August 17, 2017

Estimation of Phases for Compliant Motion

Tesfamichael Marikos Hagos

School of Electrical Engineering

Thesis submitted for examination for the degree of Master of Science in Technology.

Espoo

Thesis supervisor:

Prof. Ville Kyrki (Aalto University)

Prof. George Nikolakopoulos (Luleå University of Technology)

Thesis advisor:

Markku Suomalainen (MSc.)

Aalto University
School of Electrical
Engineering

AALTO UNIVERSITY
SCHOOL OF ELECTRICAL ENGINEERING

ABSTRACT OF THE
MASTER'S THESIS

Author: Tesfamichael Marikos Hagos

Title: Estimation of Phases for Compliant Motion

Date: August 17, 2017: Language: English Number of pages: 9+59

Department of Electrical Engineering and Automation

Professorship: Automation technology

Supervisor: Prof. Ville Kyrki

Advisor: Markku Suomalainen (MSc.)

Nowadays adding a skill to the robot that can interact with the environment is the primary goal of many researchers. The intelligence of the robot can be achieved by segmenting the manipulation task into phases which are subgoals of the task and identifying the transition between them.
This thesis proposes an approach for predicting the number of phases of a compliant motion based manipulation task and estimating their corresponding HMM model that best fit with each segmented phase of the task. Also, it addresses the problem of phase transition monitoring by using recorded data. The captured data is utilized for the building an HMM model, and in the framework of task segmentation, the phase transition addressed. In this thesis, the concept of non-homogeneous HMM is used in modeling the manipulation task, wherein hidden phase depends on observed effect of performing an action (force). The expectation-maximization (EM) algorithm employed in estimating the parameters of the HMM model. The EM algorithm guarantees the estimation of the optimal parameters for each phase of the manipulation task. Hence the modeling accuracy of the forced based transition is significantly enhanced compared to position based transition. To see the performance of the phase transition detection a Viterbi algorithm was implemented. A Cartesian impedance controller defined by [6] for each phase detected is used to reproduce the learned task. The proposed approach is investigated with a KUKA LWR4+ arm in two test setups: in the first, we use parameter estimation for a single demonstration with three phases, and in the second experiment, we find a generalization of the parameter estimation for multiple demonstrations. For both experiments, the transition between phases of the manipulation task is identified. We conclude that our method provides a convenient platform for modeling and estimating of model parameters for phases of manipulation task from single and double demonstrations.

Keywords: Non-homogenous hidden markov model, Learning from Demonstration, Multi-class regression, Gradient decsent

Preface

First, all I would like to express my special thanks to my supervisor Professor Ville Kyrki as well as my advisor Markku Suomalainen for allowing me to work on this research title with the motivated, passionate and friendly group of people. Working on this research topic helped me on knowing so many new things, and I am thankful to them. I would like to thank Professor Ville Kyrki for his continuous guidance and patience from the beginning of the project until the end. In addition to his support a detailed explanation of the in-depth concept of the research topic, I have learned how to work and interact with in a group. My deepest gratitude goes to Markku for guiding me throughout the project to reach the initial objectives and continuous support afterward by motivating me all through the work and his guidance to improve the language and style of this thesis.

Researching with the Intelligent Robotics group was an opportunity I will never forget and that allowed me to meet awesome people. Especially thanks to the support of my neighbors and friend Mattia. My best wishes for his future, I am sure he will succeed in every project he manages to start.

I would also want to thank Space Master coordinators for giving me the chance to study in this program and make me one of the few chosen people for the scholarship opportunity. I would also want to thank my examiner for the Master thesis from LTU's side Prof. George Nikolakopoulos for his fast response to my emails when I need him the most.

I also want to thank my Space Master colleagues, especially those who stood together with me during second year of the program in Aalto University.

Finally, I would like to thank intelligent robotics group members during my thesis work period and my parents for their continuous support.

August 17, 2017 Tesfamichael Marikos Hagos

Contents

Symbols and abbreviations

Symbols

A	State transition probability distribution (Transition matrix)
B	Observation probability distribution
A_j	State matrix corresponding to phase j
B_j	Effect of action matrix corresponding to phase j
$\mathbf{a}_t$	Effect of performing an action at time t
D	Number of demonstrations
F_{cmd}	Superposed force in the Cartesian impedance controller
F_k	Force derived by the spring system in the Cartesian impedance control law
K	Dimensionality of a demonstration
k_c	Stiffness coefficients of the Cartesian impedance controller
M	Number of observation symbols (discrete case) or densities composing the observation set mixture (continuous case)
N	Number of states (Hidden Markov Model (HMM))
$\mathbf{s}_t$	Observed state at time t (HMM)
w_j	Weight matrix corresponding to phase j
z_t	Combined observed state and effect of performing action at time t (HMM)
$\alpha_t(i)$	Forward variable defined in Section 4.3.1
$\beta_t(i)$	Backward variable defined in Section 4.3.1
$\gamma_t(i)$	Marginal likelihood defined in Section 4.3.1
π	State priors (HMM)
ρ_t	Hidden phase at time t (HMM)
Σ_j	Covariance matrix corresponding to phase j
θ	Parameters of the HMM
$\zeta_t(i,j)$	Joint distribution defined in Section 4.3.1

Operators

$\frac{\mathrm{d}}{\mathrm{d}t}$	derivative with respect to variable t
$\frac{\partial}{\partial t}$	partial derivative with respect to variable t
$\sum_i$	sum over index i
$A \cdot B$	product of matrices A and B

Abbreviations

AIC	Akaike Information Criterion
ARHMM	Auto-Regressive Hidden Markov Model
BIC	Bayesian Information Criterion
BN	Bayesian Network
EM	Expectation Maximization
ETAR	Effect-of-action-based Transition Auto-regressive hidden Markov model
FRI	Fast Research Interface
GD	Gradient Descent
HMM	Hidden Markov Model
KL	Kullback-Leibler divergence
KT	Kinesthetic Teaching
KRC	KUKA Robot Controller
KRL	KUKA Robot Language
LfD	Learning from Demonstration
LR	Logistic Regression
MM	Markov Model
MN	Markovian Network
MP	Markov Process
NHMM	Non-homogeneous Hidden Markov Model
OL	Observational Learning

Orocos	Open robot control software
pdf	Probability Density Function
ROS	Robot Operating System
SP	Starting Position
STAR	State-based Transition Auto-regressive hidden Markov model
TO	Teleoperation Teaching

List of Figures

1 Introduction

Nowadays the usage of robots for industrial applications is increasing, but the usage of such robots is limited to structured environments. To use robots in the unknown environment, they must be intelligent, and flexible that can interact with their surroundings. Such intelligence and flexibility can be achieved by using additional sensors such as cameras, force/torque sensors or laser distance sensors. Force/torque controlled robots are efficient systems in assembly task due to their position uncertainty handling capability.

Assembly tasks is a combination of unconstrained motion and constrained motion which involves contact with an unknown environment. The use of position control alone for the contact phase of the assembly task can cause an unlimited rise of the contact force, ultimately leading to behaviors such as breakage or unstable control [1]. Such tasks are called compliant motion i.e. "manipulation tasks which involve contact between the tool and the environment, and during the execution of which the end-effector movement is modified by the contact forces" [1]. In general, contact forces are important for the correct execution of the assembly task. The structure of the unknown environment can be used as guidance for the correct execution, by using an impedance controller along the task.

The compliant motion realized by force-based control is called active compliant motion. Active compliant motion can sufficiently solve the wider variety of problems in complex situations compared to its counterpart passive solution which uses compliant gripper which is an application specific. Such problems associated with the tool in contact can be regarding the contact configurations and contact states.

In the context of compliant motion tasks, transferring a skill to a robot and making it adaptable to the environment is a complicated process. One way of automating assembly tasks is modeling the assembly task of the robot itself; by adding the required skills to the robot that make it intelligent by making observe and interact with its surrounding. In this context, learning mechanisms are needed to acquire knowledge from such interaction with the environment. Learning from demonstration (LfD) is one approach in which a robot can learn from human examples.

LfD is a technique based on the direct demonstration of a task to a robot. The teaching process can be achieved using teleoperation teaching as in [3], kinesthetic teaching as in [4], or by observational teaching as in [5]. Kinesthetic teaching technique is easy for small and light weight robots and results in relatively successful demonstrations. LfD depends on

1. an environment setup to obtain data about a task from one or more demonstrations;

2. a methodology to learn the probabilistic model from the recorded data;

3. a technique for converting the learned model into robot's actions that can reproduce the task.

During the demonstration phase, related to point 1) the required data is recorded. Inspired by [6] and [7], in this thesis, our primary concern is with point 2) applied

to compliant motion tasks in which collected positions and forces are used for probabilistic modeling and segmentation of manipulation task.

This thesis presents a contribution to LfD, in the context of compliant motion based tasks. The biggest challenges in automatic translation a compliant motion demonstration into an executable robot program that can repeat the same task are:

1. to find the number of phases which are subgoals of the task
2. to estimate model parameters of each phase
3. reproduction phase change detection

In assembly task, inspired by the relevance of contact force, the primary target of this thesis is to study and analyze probabilistic modeling and segmentation of phases of manipulation task. In LfD paradigm this is achieved by using the force data as a feature vector in the hidden phase transition HMM model of the manipulation task. The model learning is achieved by creating a generalized HMM model from one or more demonstrations for the particular task [7]. The controller defined in [6] is used to reproduce the learned compliant motion task.

This thesis aims to: (1) kinesthetically teach a robot a compliant motion task which involves multiple phases. (2) estimate the phases and their corresponding model parameters from the data obtained from the demonstration. (3) reproduce the manipulation task using the model parameter estimated and a controller defined by [6], i.e. an impedance controller defined by the desired direction of movement, the number of compliant axes and their directions for each estimated phases.

To reach this objective, we conducted experiments with a KUKA LWR4+ arm in two test setups, one for manipulation task starting from a single starting position and another similar task with two different starting position. Based on the setup, a kinesthetic teaching approach is used to teach the robot a three phase manipulation task and the corresponding position and force are recorded. Matlab is used to estimate the model parameters for each phase using soft-max based expectation-maximization (EM) algorithm from the recorded data. Finally, during reproduction stage, an HMM Viterbi algorithm is used to detect the phase of the manipulation task based on the online reading of the position and force. And based on the estimated hidden phase sequence an impedance controller assigned to each phase of the task is activated. The impedance controller is defined by the desired direction of movement, compliant axes, and their directions.

The main contributions are modeling and parameter estimation of a manipulation tasks for a robot based on LfD paradigm. There has been active research in LfD for decades, mostly addressing how tasks are learned by kinesthetic teaching. In this thesis, we explore how to use HMM probabilistic encoding approaches for modeling observations and state transition uncertainties of the manipulation task as used in [7]. The probabilistic model of the manipulation task has two aims: i) estimate the number of hidden phases of the manipulation; ii) estimate the model parameters corresponding to each phase. In addition to the modeling of the manipulation, the impedance control framework developed in [6] is used in order to reproduce the manipulation task.

Keeping in mind that our approach can be applied for a task with multiple phases, we test our methodology with manipulation tasks which consist of three phases. First, a human teacher kinesthetically demonstrates the manipulation task and the position and effect of performing an action i.e. force are recorded. Using the recorded data the number of phases and their corresponding parameters are estimated. In addition to model parameter estimation, we use the technique used in [6] to identify the desired direction of movement, the number of compliant axes and their directions. We use this information to construct an impedance controller for each phase which can reproduce the assembly motion despite uncertainty in the starting position.

The thesis is structured as follows. In Section 2 we review some relevant research (and explain why our method is necessary). In Section 3, we introduce the basics of Markov models and HMM models. Section 3, presents the parameters of the standard HMM model definition, three problems that can be formulated within HMMs, and their corresponding solutions. The comparison between generative and discriminative HMMs is also part of Section 3. Section 4 introduces parameters used to describe the model with their corresponding mathematical description. The algorithm used for estimating parameters of the model from single and multiple demonstrations are also part of this section. Section 5 presents hardware and software architecture used in experiments. Section 6 discusses the experimental setup and the results obtained. Finally, Section 7 concludes the work with a summary.

2 Learning from Demonstration

A major challenge in robotics is automating assembly tasks in unstructured environments. As proposed in [8] and [9], learning is a good approach for robots working in dynamic environments. Teaching a robot a skill can be performed using direct interaction with an environment or using demonstrations carried out by an operator (which can be a human teacher or another robot) which is commonly known as learning from demonstration (LfD) or imitation learning.

Consider a manipulation task (such as an assembly task) which has many phases, as described in [10]. Such a task can be segmented into a series of finite phases. For example, for a task that involves approaching and sliding across the environment, the first stage corresponds to unconstrained motion in which the tool moves freely in the air. The subsequent phase begins when the tool is in contact with the environment, i.e. constrained motion, which can be executed by an impedance controller defined by a direction of movement, number of compliant axes and their corresponding direction. The transition between phases corresponds to events such as non-contact and in-contact of the tool with the environment [7] that correspond the system dynamics change as the result robot action. As shown in the approaching and sliding case, the shift between subgoals of the task is represented by a shift of robot's controller from position to Cartesian impedance.

Here the main problem to be addressed (especially when we have multiple constrained motion phases) is how to estimate the number of phases of the manipulation task and how to move our tool within each phase so that we can reach the desired position in an unknown environment. One approach to avoid the application of large force due to position uncertainty is to detect phases of the manipulation task based on the contact force and assign an impedance controller as in [6] for each estimated phases.

Since programming a robot is a complex process, one way to simplify the process is to teach the task and its corresponding model parameters from the demonstration. This section mainly offers an overview of how the teaching, modeling and reproduction work in LfD paradigm.

LfD based paradigm involves the following three steps [3]:

1. Teaching
2. Learning
3. Reproduction

2.1 Teaching

Transferring of skills to a robot can be done by three methods that help us to perform programming by demonstration. Those are teleoperation, kinesthetic teaching, and observational learning [2].

Teleoperation (TO) uses a master manipulator like a joystick or haptic gloves for teaching and a slave manipulator for execution. A human teacher controls the

master manipulator. In teaching the pouring skill [3], a haptic device was used to record both force and position data. The biggest problem with TO is the teaching process is dependent the experience of the demonstrator [11]. The complexity of Teleoperating teaching increases with an increasing number of DOF (especially when the robot has more than 6 DOF) [3].

In Kinesthetic Teaching (KT), is based on physically maneuvered of the robot through the task by the human. Gravity compensation mode of the robot activation is required for easy control of the robot. KT is useful for small and lightweight robots and is more reliable for teaching complex in contact tasks [12], [13].

The third type of teaching is observational learning (OL), in which sensors attached to the human operator are used to record the actions of the demonstrator. S. Calinon and A. Billard [14] uses an active OL method that put the human in the loop of the robot's learning process, by letting the robot to observe the task performed by the teacher through a motion sensor and then using kinesthetic teaching for progressive refinement. Even though such learning process is natural, it is not efficient for assembly applications because it requires many sensors such as data gloves, magnetic trackers and stereo vision for collecting data from the demonstration.

2.2 Learning

Learning is the second stage of the LfD paradigm. And involves segmenting and finding a model of the segmented task. Segmenting phases of a manipulation task can be achieved by the non-probabilistic or probabilistic approaches. The non-probabilistic approach exploits geometric knowledge of the contacted objects that relies on force and torque data to estimate force component and contact location [19], [20]. The proposed approach suffers from poor stochastic foundation which results in an inaccurate modeling of the task due to sensor noise and slow phase transitions [23]. The probabilistic approach (more specifically HMM-based) which has a stochastic basis and it is characterized with fast phase transition recognition [21], [22]. And also overcome the problem of inaccurate modeling by using the stochastic variables mean and variance incorporated with HMM.

There has been considerable work on segmenting a manipulation task into subtasks. [23], [24] propose a Bayesian sequential Monte Carlo methods (also known as particle filters) to estimate the continuous geometry parameters and recognize the discrete contact information. In their proposal, they use a position of the contact point(s), a direction of contact normal(s), as geometry parameters and contact formation as discrete. Similarly, [25] proposed a Bayesian approach for state estimation and monitoring the contact state transition for sensor-based robot tasks. The main problem of [23] - [25] is the requirement of state graph for all possible contact formations. In [26] impedance controller parameters (such as stiffness, damping, and inertia forces) are learned by fitting dynamic equation using weighted least square. [28] uses supervised learning algorithm (stochastic gradient boosting algorithm) to perform segmentation. The algorithm does not depend on the geometric model, but it requires training example.

Hidden Markov models based modeling of contact formation for compliant motion

robots have successfully used for the opening the door in [21], [29] - [32]. Relevant related previous works assume predefined phases of the manipulation task [29], [33], [34]. Romano [34] and Johansson [35] use a human-inspired controller for grasping objects in which a lower level controller represents each grasping phases. In the proposed framework tactile events are used to detect the transitions between phases. Andrews [33] also proposes a controller for performing in-hand manipulation organizing a problem into defined three phases. For given contact state network and description of the states T. Debus ([29]) propose an estimate of the contact state for the peg-in-hole task using an HMM. The primary target of this thesis is to learn a probabilistic model (specifically a time-varying HMM) of the task demonstrated and the detailed description of the model will be clarified in Section 4.

2.3 Reproduction

The third and the final step of the LfD paradigm is the reproduction step. Assuming that we have a model learned from the demonstrated manipulation task, many research papers have proposed ways of reproducing the learned model. In [26] transient behavior during reproduction results in a problem in detecting the transition between the segmented subtasks. In [27], an active control strategy learned from the position and force profile with variable stiffness is proposed to reproduce. The skill learned is based kinesthetic teaching for the positional profile and haptic device for the force profile.

In the context of reproduction, this thesis is mainly a continuation of [6]. Therefore after segmentation of phases an impedance controller is assigned for each segmented phase. Finally, a Viterbi algorithm is used to detect the phases of the manipulation task in which it activates a controller specifically defined for the currently detected phase.

3 Hidden Markov Models

This section describes Markov chain and the standard Hidden Markov Models (HMMs). We start by defining Markov chain, then HMMs and the problems involved with HMMs and their corresponding solutions. The basic idea behind the generative and discriminative HMM models followed by their comparison.

3.1 Markov Model

Markov models are a stochastic model used for randomly changing systems. Markov models are models in which future states depend only on the current state but not on the whole history. For a set of distinct states $\rho = \{\rho_1, \rho_2, \cdots, \rho_N\}$, the system moves to one of the states according to a set of state transition probabilities A at each discrete time step t.

For the first order Markov chain in Figure 1, the next state only depend on the current state.

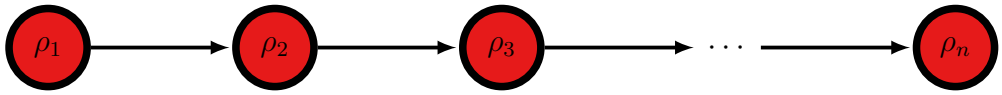


Figure 1: First order Markov Chain

For the first-order Markov chain shown in Figure 1, the edges represent the transition probabilities and the dependency on the previous state is given by conditional probability

$$p(\rho_n|\rho_1, \cdots, \rho_{n-1}) = p(\rho_n|\rho_{n-1}) \tag{1}$$

and the joint probability of a sequence of N observations is given by

$$p(\rho_1, \cdots, \rho_N) = \prod_{n=1}^{N} p(\rho_n|\rho_1, \cdots, \rho_{n-1}) = p(\rho_1) \prod_{n=2}^{N} p(\rho_n|\rho_{n-1}) \tag{2}$$

3.2 HMM Model Parameter Definition

The Markov model presented in Section 3.1 has limited power in many applications. An extended version which is called Hidden Markov Model (HMM) is required to increase the model's representation power. The basic idea with HMM is that we do not know what generates the observation sequence.

For a hidden state sequence set $\rho = \{\rho_1, \rho_2, \cdots, \rho_n\}$ and for observed sequence set $Z = \{z_1, z_2, \cdots, z_n\}$ a generative (standard) hidden Markov model is described in Figure 2

A standard HMM is defined by three parameters $\theta = \{A, B, \pi\}$, where A is the hidden state transition probability matrix, B is the emission probability matrix assuming discrete observation, and π is the initial probability [36].

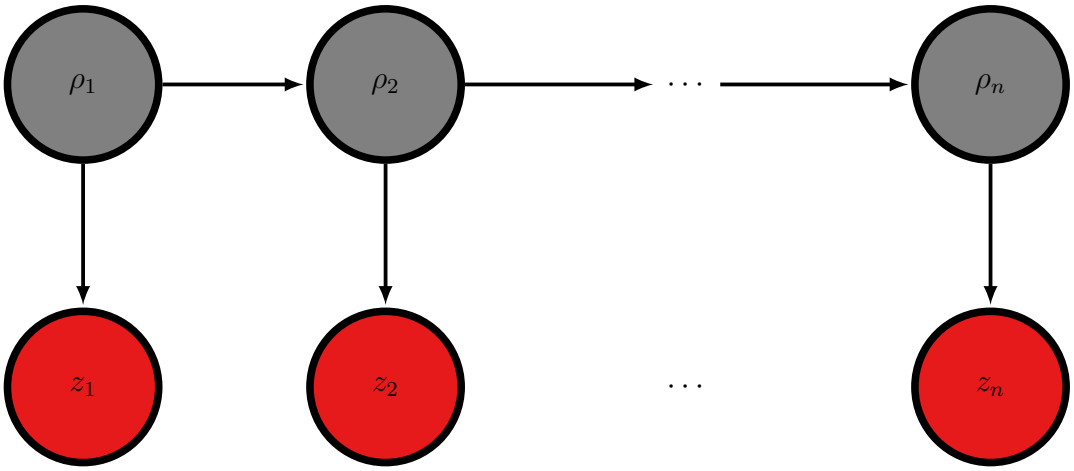


Figure 2: Standard (generative) HMM model

1. Transition probability, $A = [a_{i,j}]$ is a transition between latent variable that describes the probability of going from state ρ_i to state ρ_j

$$a_{i,j} = p(\rho_t = j | \rho_{t-1} = i), \quad i, j = 1, 2, \cdots, N \tag{3}$$

2. Initial probabilities $\pi = [\pi_i], i = 1, 2, \cdots, N$ where $\pi_i = p(\rho_1 = i)$, describes the probability of state i being the initial state.

3. Observation (Emission) probability distribution, $p(z_t|\rho_t = i), i = 1, 2, \cdots, N$, which is the conditional distribution of the observation variable from specific state. Depending on the distribution of the emission (observation probability), the HMM model can be classified as a discrete or continuous HMM model

   (a) Discrete: If z_t is discrete, the distribution associated with each hidden state specifies the probability of emitting each observable

$$b_i(k) = p(z_t|\rho_t = i), \quad i, j = 1, 2, \cdots, N \tag{4}$$

   (b) Continuous: if the observation probability is continuous, then the parameters, b_i, are defined by Gaussian distributions.
   The multivariate M mixture of Gaussian distribution for vector valued observations $z_t \in \mathbb{R}^D$ is given by

$$b_i(z_t) = \sum_{m=1}^{M} \frac{c_{i,m}}{(2\pi)^{\frac{D}{2}} |\Sigma_{i,m}|^{\frac{1}{2}}} \exp\left[-\frac{1}{2}(z_t - \mu_{i,m})^T \Sigma_{i,m}^{-1} (z_t - \mu_{i,m})\right] \tag{5}$$

   which is commonly called a Gaussian-mixture HMM. The parameter set of the PDF Σ_i comprises scalar weigts $c_{i,m}$, Gaussian mean vectors, $\mu_{i,m} \in \mathbb{R}^D$, and Gaussian covariance matrices, $\Sigma_{i,m} \in \mathbb{R}^{D \times D}$
   For a single multivariate model the state-dependent output PDF becomes a unimodal Gaussian

$$b_i(z_t) = \frac{1}{(2\pi)^{\frac{D}{2}}|\Sigma_{i,m}|^{\frac{1}{2}}} \exp\left[-\frac{1}{2}(z_t - \mu_{i,m})^T \Sigma_{i,m}^{-1}(z_t - \mu_{i,m})\right] \quad (6)$$

the unimodal Gaussian is commonly called (continuous-density) Gaussian HMM.

3.3 HMM Problem Definition and Solution

There are three standard problems that can be formulated within HMMs [36]. Those three problems need to be solved before applying HMMs for a wide area of applications are:

1. **Evaluation problem:** given the sequence of observations $\mathbf{z} = z_1, z_2, \cdots, z_N$ and the HMM model θ, compute the likelihood $p(\mathbf{z}|\theta)$
2. **Decoding problem:** given the sequence of observations $\mathbf{z} = z_1, z_2, \cdots, z_N$ and the HMM model θ, infer the most likely hidden state sequence
3. **Learning problem:** given the observation sequence $\mathbf{z} = z_1, z_2, \cdots, z_N$, estimate the parameters of the HMM model θ.

For a standard (generative) HMM model defined in Figure 2, the solution of the three basic HMM problem is described in the following sections.

3.3.1 Solution to Evaluation Problem: Computing Likelihoods

Given a sequence of observations $\mathbf{z} = z_1, z_2, \cdots, z_T$ and a HMM with complete model parameter $\theta = (A, B, \pi)$ the main target of the evaluation problem is to evaluate how well the model predicts the given observation sequence i.e. likelihood $p(\mathbf{z}|\rho, \theta)$. From the definition of emission probability B for state sequence $\rho = \{\rho_1, \rho_2, \cdots, \rho_T\}$, we have

$$p(\mathbf{z}|\rho, \theta) = \prod_{t=1}^{N} p(z_t|\rho_t, \theta) = b_{\rho_1}(z_2) b_{\rho_2}(z_2) \cdots b_{\rho_T}(z_T) \quad (7)$$

and from the definition of initial distribution π and hidden state transition A

$$p(\rho|\theta) = \pi_{\rho_1} a_{\rho_1,\rho_2} a_{\rho_2,\rho_3} \cdots a_{\rho_{T-1},\rho_T} \quad (8)$$

the likelihood of the observed sequence can be obtained by summing the joint probability of the hidden states sequence ρ and the observation sequence $\mathbf{z}$ over the hidden states

$$\begin{aligned} p(\mathbf{z}|\theta) &= \sum_{\rho} p(\mathbf{z}, \rho|\theta) \\ &= \sum_{\rho} p(\mathbf{z}|\rho, \theta) p(\rho|\theta) \\ &= \sum_{\rho} \pi_{\rho_1} b_{\rho_1} a_{\rho_1,\rho_2}(z_2) b_{\rho_2}(z_2) a_{\rho_2,\rho_3} \cdots b_{\rho_N}(z_T) a_{\rho_{N-1},\rho_N} \end{aligned} \quad (9)$$

Evaluating Equation 9 directly results in exponential complexity with respect to the length of observations T. A better solution is to compute the joint probability

distribution of the hidden state sequence ρ and the observation sequence $\mathbf{z}$ using the forward algorithm [36]- [41]. The forward probability at time t is given by

$$\alpha_t(i) = p(z_1, z_2, \cdots, z_t, \rho_t = i|\theta) \tag{10}$$

The forward algorithm is an efficient solution for the evaluation problem and can be recursively computed as follows [37]

1. Initialization: for $i = 1, \cdots, N$

$$\alpha_1(i) = \pi_i b_i(z_1) \tag{11}$$

2. Induction: for $j = 1, \cdots, N$ and $t = 1, \cdots, T-1$

$$\alpha_{t+1}(j) = \left(\sum_{i=1}^{N} \alpha_t(i) a_{ij}\right) b_j(z_{t+1}) \tag{12}$$

3. Evaluating the probability:

$$p(\mathbf{z}|\theta) = \sum_{i=1}^{N} \alpha_T(i) \tag{13}$$

Now the complexity of the forward algorithm becomes linear with T unlike the direct calculation mentioned earlier, which had an exponential complexity. The pseudo code representation of the forward algorithm is shown in Algorithm 1.

Algorithm 1 Forward Algorithm

function FORWARD($\mathbf{z}, \theta$)
  for $i = 1$ to N **do** ▷ Initialization
    $\alpha_1(i) = \pi_i b_i(z_1)$
  end for
  for $j = 1$ to N **do** ▷ Induction
    for $t = 1$ to $T-1$ **do**
      $\alpha_{t+1}(j) = \left(\sum_{i=1}^{N} \alpha_t(i) a_{ij}\right) b_j(z_{t+1})$
    end for
  end for
  $p(\mathbf{z}|\theta) = \sum_{i=1}^{N} \alpha_T(i)$ ▷ Evaluating the probability
end function

3.3.2 Solution to Decoding Problem: Decoding Hidden States

For the observation sequence and the HMM model $\theta = (A, B, \pi)$ given, the aim of the decoding problem is to estimate the hidden state sequence that most likely results the observation sequence. There are two solutions for the decoding problem, one based on the forward-backward algorithm and the other on the Viterbi algorithm.

a). Forward-Backward Algorithm

The state sequence for any point in time can be estimated using the forward-backward algorithm. The algorithm is based on a two-step dynamic programming. The first one goes forward in time computing the probability of ending up in any particular state $p(\rho_t = j|\mathbf{z}, j)$ which is called the forward algorithm as described in 3.3.1, while the second goes backward in time computing the probability of observations given a starting point $\rho_t = j$: $p(z_{t+1}, \cdots, z_T|\rho_t = j)$ which is called backward algorithm.

The backward algorithm starts at the end and works backward to the beginning and is defined as

$$\beta_t(i) = p(z_{t+1}, z_{t+2}, \cdots, z_T|\rho_t = i, \theta) \tag{14}$$

Similarly as the forward algorithm, the backward algorithm can also be computed recursively as follows [37]:

1. Initialization: for $i = 1, \cdots, N$

$$\beta_T(i) = 1 \tag{15}$$

2. Induction: for $t = T - 1, \cdots, 1$ and or $i = 1, \cdots, N$

$$\beta_t(i) = \sum_{j=1}^{N} a_{ij} b_j(z_{t+1}) \beta_{t+1}(j) \tag{16}$$

3. The state sequence at time t can be calculated by computing the maximum of the smoothed variable $\max \gamma_t(i)$, which is the normalized product of the forward variable α and the backward variable β,

$$\gamma_t(i) = p(\rho_t = i|\mathbf{z}, \theta) = \frac{\alpha_t(i)\beta_t(i)}{p(\mathbf{z}|\theta)} \tag{17}$$

Pseudo code representation of the backward algorithm is shown in Algorithm 2. Pseudo code representation of the forward-backward algorithm is shown in Algorithm 3.

b). Viterbi Algorithm

Viterbi algorithm is an algorithm which maintains the highest probability paths at each possible state instead of a list of all possible paths. The algorithm is structurally quite similar to the forward algorithm that makes uses of a dynamic programming trellis. Assuming the probability being in state j is represented by cell of the Viterbi trellis $v_t(j)$, the most probable state sequence is given by [36], [38]

$$v_t(j) = \max_{\rho_1, \cdots, \rho_{t-1}} p(z_1 \cdots z_t, \rho_1, \cdots, \rho_t = j) \tag{18}$$

Algorithm 2 Backward Algorithm

function BACKWARD($\mathbf{z}, \theta$)
  for $i = 1$ to N **do** ▷ Initialization
    $\beta_T(i) = 1$
  end for
  for $j = T - 1$ to 1 **do** ▷ Induction
    for $t = 1$ to N **do**
      $\beta_t(i) = \sum_{j=1}^{N} a_{ij} b_j(z_{t+1}) \beta_{t+1}(j)$
    end for
  end for
end function

Algorithm 3 Forward Backward Algorithm

function FORWARDBACKWARD($\mathbf{z}, \theta$)
  for $i = 1$ to N **do** ▷ Initialization of first forward message
    $\alpha_1(i) = \pi_i b_i(z_1)$
  end for
  for $j = 1$ to N **do** ▷ Induction
    for $t = 1$ to $T - 1$ **do**
      $\alpha_{t+1}(j) = \left(\sum_{i=1}^{N} \alpha_t(i) a_{ij} \right) b_j(z_{t+1})$
    end for
  end for
  for $i = 1$ to N **do** ▷ Initialization of last backward message
    $\beta_T(i) = 1$
  end for
  for $j = T - 1$ to 1 **do** ▷ Induction
    for $t = 1$ to N **do**
      $\beta_t(i) = \sum_{j=1}^{N} a_{ij} b_j(z_{t+1}) \beta_{t+1}(j)$
    end for
  end for
  $\gamma_t(i) = p(\rho_t = i | \mathbf{z}, \theta) = \frac{\alpha_t(i)\beta_t(i)}{p(\mathbf{z}|\theta)}$ ▷ Most likely state sequence at time t
end function

The recursive computation of Viterbi algorithm is based on updating each trellis cell from the previously computed cell [38] and is given by

$$v_t(j) = \max_{i=1}^{N} v_{t-1}(i) a_{ij} b_j(z_t) \tag{19}$$

Unlike the forward algorithm which computes the sum of probability, the Viterbi algorithm calculates the maximum of the probability. To find the most probable path, backtracking pointer $\beta_t(j)$ which points from state j at time t to state i at time $t - 1$ is required [38]. The pseudo code representation of the Viterbi algorithm is shown in Algorithm 4

Algorithm 4 Viterbi Algorithm

function FORWARD($\mathbf{z}, \theta$)
  Path:={}
  $\beta := \{\}$
  for $i = 1$ to N **do** ▷ Initialization
    $v_1(i) = \pi_i b_i(z_1)$
    $\beta_1(i) = 0$
  end for
  for $t = 1$ to $T - 1$ **do** ▷ Induction
    for $j = 1$ to N **do**
      $v_{t+1}(j) = \max_{j'=1}^{N} \left(v_t(j') a_{j'j}\right) b_j(z_{t+1})$
      $\beta_{t+1} = \arg\max_{j'=1}^{N} \left(v_t(j') a_{j'j}\right)$
    end for
  end for
  Append $\beta_{\hat{j}}$ to β
  Append $\rho_{\hat{j}}$ to Path
  return Path
end function

3.3.3 Solution to Learning Problem:Adjustment of the Model Parameters

The main concern of the learning problem is adjusting the parameters of an HMM given a set of observation sequences and initial guess of the model parameters using the maximum likelihood as optimization criteria. In order to find the model that best fits a given observation sequence, an Expectation-Maximization (EM) algorithm is used. EM algorithm, when applied to HMM, is called Baum-Welch algorithm 3.3.1 and the basic idea behind the algorithm is provided in Appendix-B. This algorithm makes use of the forward-backward algorithm in order to calculate intermediate variables which are used in order to re-estimate the model parameters. The intermediate variables are

1. $\gamma_t(i)$, the conditional hidden state probability can be calculated in terms of the forward and backward messages obtained in Section 3.3.1

$$\gamma_t(i) = p(\rho_t = i|\mathbf{z}, \theta) = \frac{\alpha_t(i)\beta_t(i)}{p(\mathbf{z}|\theta)} \tag{20}$$

   The summation of $\gamma_t(i)$ over t represents the transition probability given the model parameters and the observation sequence $\mathbf{z}$ [36], [37].

2. $\zeta_t(i, j)$, the joint probability of being in state i at time t and in state j at time $t + 1$ that is a joint probability defined interms of the forward and backward variables of Section 3.3.1.

$$\zeta_t(i, j) = p(\rho_t = i, \rho_{t+1} = j|\mathbf{z}, \theta) = \frac{\alpha_t(i) a_{ij} b_j(z_{t+1}) \beta_{t+1}(j))}{p(\mathbf{z}|\theta)} \tag{21}$$

The summation of ζ_t over t represents the expected number of transitions from state i to state j given the model parameters and the observation sequences $\mathbf{z}$ [36], [37].

The update of the model parameters is obtained from the intermediate variables defined as the transition probability is represented by the normalized summation of $\gamma_t(i)$ over t and the emission probability is the normalized summation of ζ_t over t. The algorithm for the re-estimation of model parameters is given by

1. for $i = 1, \cdots, N$, let
$$\pi_i = \gamma_1(i) \tag{22}$$
2. for $i = 1, \cdots, N$ and $j = 1, \cdots, N$
$$a_{ij} = \frac{\sum_{t=1}^{T-1} \zeta_t(i,j)}{\sum_{t=1}^{T-1} \gamma_t(i)} \tag{23}$$
3. for $j = 1, \cdots, N$ and $k = 1, \cdots, T$
$$b_j(k) = \frac{\sum_{t \in \{1,\cdots,T\}, z_t = k} \zeta_t}{\sum_{t=1}^{T} \gamma_t(i)} \tag{24}$$

Pseudo code representation of the EM algorthm is shown in Algorithm 5

3.4 Generative and Discriminative HMM

HMMs are models that are used to model the joint distribution. Depending on how the joint probability distributions of the hidden and the observed variable are defined $p(\mathbf{z}, \rho)$, HMMs can be classified as generative or discriminative HMMs [39]- [41].

3.4.1 Generative Hidden Markov Model

As described in Section 3.3, the joint probability of generative HMMs is decouples as $p(\mathbf{z}, \rho) = p(\mathbf{z}|\rho)p(\rho)$. Such type of joint distributions computation requires explicit model of $p(\rho)$, which is complex. The dependencies between the hidden and observed are only from the former to the later: the hidden variable generates the observable variable [39]- [41].

The problem of classification in a generative HMMs, i.e. predicting a discrete hidden phases variable $\rho = \{\rho_1, \rho_2, \cdots, \rho_N\}$ given the a vector of observations $\mathbf{z} = \{z_1, z_2, \cdots, z_N\}$ is modeled based the joint distribution $p(\mathbf{z}, \rho)$. To model the joint distribution $p(\mathbf{z}, \rho)$, a generative HMM makes use the methodlogy described in Section 3.3. That is, the joint probability of a hidden state sequence ρ and an observation $\mathbf{z}$ factorized as

$$p(\mathbf{z}, \rho) = p(\rho_1) \prod_{t=2}^{T} p(\rho_t|\rho_{t-1}) p(z_t|\rho_t) \tag{25}$$

Algorithm 5 EM algorithm

function EM($\mathbf{z}, \theta$)

E-step

FORWARDBACKWARD($\mathbf{z}, \theta$) ▷ Algorithim 3

$\gamma_t(i) = p(\rho_t = i|\mathbf{z}, \theta) = \frac{\alpha_t(i)\beta_t(i)}{p(\mathbf{z}|\theta)}$

$\zeta_t = p(\rho_t = i, \rho_{t+1} = j|\mathbf{z}, \theta) = \frac{\alpha_t(i)a_{ij}b_j(z_{t+1})\beta_{t+1}(j))}{p(\mathbf{z}|\theta)}$

M-Step

for $i = 1$ to N **do** ▷ Initial distribution estimate

  $\pi_i = \gamma_1(i)$

end for

for $i = 1$ to N **do** ▷ Transition probability re-estimate

  for $j = 1$ to N **do**

    $a_{ij} = \frac{\sum_{t=1}^{T-1} \zeta_t}{\sum_{t=1}^{T-1} \gamma_t(i)}$

  end for

end for

for $k = 1$ to T **do** ▷ Emission probability re-estimate

  for $j = 1$ to N **do**

    $b_k(j) = \frac{\sum_{t\in\{1,\cdots,T\}, z_t=k} \zeta_t}{\sum_{t=1}^{T} \gamma_t(i)}$

  end for

end for

---

end function

3.4.2 Discriminative Hidden Markov Model

If the purpose is to classify or segment data, we can directly model the conditional probability $p(\rho|\mathbf{z})$, as $p(\rho)$ model is not a requirement. The direct conditional probability modeling avoids the correlation of the observation variable $\mathbf{z}$. Such models try to model the difference or discrimination between different observed variables and they are commonly called Discriminative HMM models [39]- [41].

The discrimination based HMM for state classification can be modeled by assuming the logarithm of the conditional probability, $\log p(\rho|\mathbf{z})$, of each state, is a linear function of feature vector $\phi(\mathbf{s}_t)$. Such description of the conditional probability of the hidden states leads to multi-class logistic regression distribution function [39]- [41]

$$p(\rho_t = j|\mathbf{s}_t, \rho_{t-1} = i) = \frac{\exp(w_{ij}\phi(\mathbf{s}_t))}{\sum_k \exp(w_{ik}\phi(\mathbf{s}_t))} \tag{26}$$

where $\sum_k \exp(w_{ik}\phi(\mathbf{s}_t))$ is the normalization constant.

To model the joint distribution $p(\mathbf{z}, \rho)$, a discriminative HMM makes two independence assumptions. First, it assumes that each hidden phase ρ_t depends only on

its immediate predecessor ρ_{t-1} and current observation z_t. Second, it also assumes that each observation variables z_t depends on immediate predecessor z_{t-1} and the previous hidden state ρ_{t-1}. With these assumptions, we can specify a discriminative HMM in terms of the distributions $p(\rho_1|z_1)p(z_1)$ over initial states; the transition distribution $p(\rho_t|z_t, \rho_{t-1})$; and the observation distribution $p(z_t|z_{t-1}, \rho_t)$. That is, the joint probability of a hidden state sequence ρ and an observation $\mathbf{z}$ factorized as

$$p(\mathbf{z}, \rho) = p(z_1)p(\rho_1|z_1) \prod_{t=2}^{N} p(\rho_t|z_t, \rho_{t-1})p(z_t|z_{t-1}, \rho_t) \tag{27}$$

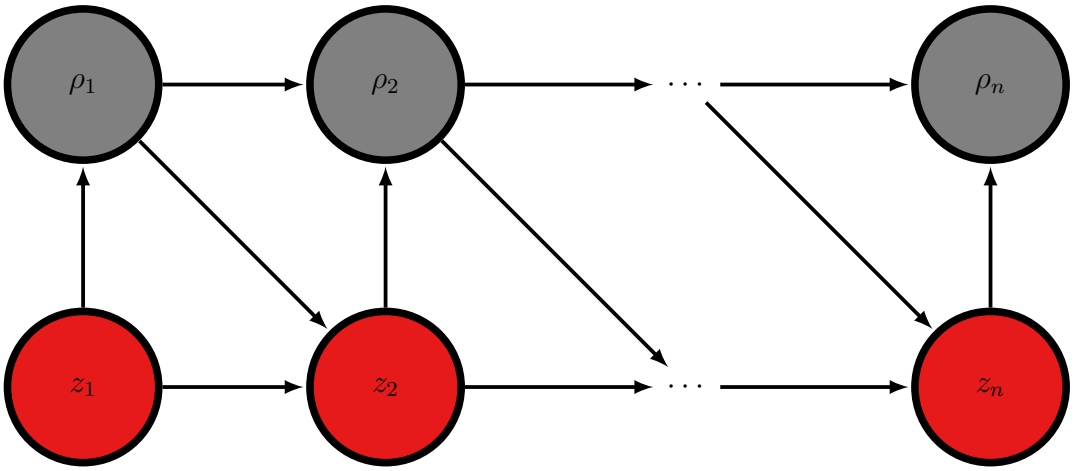


Figure 3: Discriminative Hidden Markov model

One way to build a probabilistic classifier is to create a joint model of the form $p(\mathbf{z}, \rho)$ and then condition on ρ, there by deriving $p(\mathbf{z}|\rho)$. This is called the generative approach. Another approach is to fit a model of the form $p(\rho|\mathbf{z})$ directly. This is called the discriminative approach. Both HMM models, i.e. generative and discriminative models describe distributions over $p(\mathbf{z}, \rho)$, but they work in different directions.

The main advantage of the discriminative HMM over its generative counterpart is that the conditional distribution $p(\rho|\mathbf{z})$ does not require a model for the probability of observation $p(\mathbf{z})$, which is not required for classification. It avoids the requirement of modeling $p(\mathbf{z})$ which is difficult to model due to highly dependent features.

4 Learning and Estimation of Phases using Hidden Markov model

HMM models assume homogeneous of state transition distribution. However, this assumption of stationarity of hidden state distribution does not hold for many real life applications. In this section, a time sequence model that extends HMM to time varying scenario will be introduced. The manipulation task we used for our experiments has multiple phases, and it is modeled as discrete hidden variable evolving over time similar to [7]. Each phase of the manipulation task has its corresponding model parameters. In this section, a notation used for model description and the proposed probabilistic model will be described. Here we explain the model parameter definition and the techniques used to learn model parameters of the probabilistic model.

4.1 Model Parameter Definition

Consider a state $\mathbf{s}_t \in \mathbb{R}^{3\times 1}$ representing the position of the end effector of a robot at time t. The effect of performing an action at time t, $\mathbf{a}_t \in \mathbb{R}^{4\times 1}$ to cause transition to the next state s_{t+1} represent the contact force concatenated with 1. This transition in state depends on the current hidden phase $\rho_t \in \mathbb{N}$. The standard form of these models is specified by bivariate processes $\{(\mathbf{z}, \rho)\}$ with $\{\mathbf{z}\}$ the distribution of the observed process that governs ρ being an unobserved, or hidden, finite Markov chain. The observed variable $\{\mathbf{z}\}$ is the column-wise concatenation of the observed state and contact force. The next state transition depends on the current phase which is hidden and the current effect of performing an action.

Therefore the effect of performing an action $\mathbf{a}_t$ in state $\mathbf{s}_t$ and phase ρ_t are modeled by transition probability $p(s_{t+1}|\mathbf{s}_t, \mathbf{a}_t, \rho_t)$ [7]. The state transitions $p(s_{t+1}|\mathbf{s}_t, \mathbf{a}_t, \rho_t)$ dynamic behavior humans rely on internal models [15]. The expected next state μ_{ρ_t} of each phase maps a state $\mathbf{s}_t$ and effect of performing an action $\mathbf{a}_t$ of the system to the next state s_{t+1}. In our case we assume that the next state is represented by a linear Gaussian model, $s_{t+1} = \mu_{\rho_t} + w_{\rho_t}$ similar as [7], where $w_{\rho_t} \sim \mathcal{N}(0, \Sigma_{\rho_t})$ is independent identically distributed (i.i.d.) Gaussian noise. Such model represents the transition dynamics of the robot and the distribution of the next state is represented as

$$s_{t+1} \sim \mathcal{N}(A_{\rho_t}\mathbf{s}_t + B_{\rho_t}\mathbf{a}_t, \Sigma_{\rho_t}) \tag{28}$$

where $A_i \in \mathbb{R}^{n\times n}$ is a state matrix, $B_i \in \mathbb{R}^{n\times(m+1)}$ is effect of action matrix and $\Sigma_i \in \mathbb{R}^{n\times n}$ covariance matrix corresponding to phase $\rho = i$. By multiplying the state $\mathbf{s}_t$ and the effect of performing an action $\mathbf{a}_t$ in state $\mathbf{s}_t$ with the state matrix A_{ρ_t} and effect of performing an action matrix B_{ρ_t}, respectively, the state transition function becomes linear in its arguments. Thus, the state transition of the model assumes linear system dynamics.

The model parameters A_i and B_i used for the model description are parameters used to describe the linear proportionality of the next state dynamics that are com-

pletely different from the hidden state transition matrix and observation probability matrix described in Section 3.3.

The Gaussian assumption of the dynamics of the robot for each phase is completely specified by a corresponding mean $\mu_{\rho_t} = A_{\rho_t}\mathbf{s}_t + B_{\rho_t}\mathbf{a}_t$ and a covariance Σ_{ρ_t}. The mean allows to integrate prior knowledge about the underlying observed state dynamics, and the covariance incorporates some high-level structured assumptions about the underlying function (e.g., smoothness).

We model the dynamics of the position of the end effector $\mathbf{s}_t$ through an HMM with non-homogeneous transition matrix at time t denoted by $P^{(t)} = [P_{ij}^{(t)}]$ $(p(\rho_t = j|\mathbf{s}_t, \rho_{t-1} = i))$. The latent variables $\rho_{1:N}$ represent the hidden phases that have generated the observed data.

We assume that the hidden phases ρ is represented by time-varying transition probability $p(\rho_t = j|z_t, \rho_{t-1})$, the values of which are modeled by multi-class logistic regression where the feature vector is the corresponding effect of performing an action i.e. forces. The time-varying transition probability depends on the previously hidden phase and the joint observation vector $\mathbf{z}$.

Observed state-based transitions auto-regressive hidden Markov model (STAR) proposed by Kroemer Et al. [7], has considered similar assumption except that feature vectors are considered a subset of the full state vector. Our approach is different in two ways. First, we use the forces as a feature vector which changes the assumption made by [7] (state-based transitions (STAR)) to effect-of-action based transitions auto-regressive hidden Markov model (ETAR). Secondly using the learned model an impedance controller is defined for each estimated phase as in [6] to reproduce the manipulation task.

The algorithm we propose has the following key points. First, the current phase depends on previous phase which allows the model to represent effects such as hysteresis [7]. Second, the Markovian property of the hidden phase and dependency on the current state allows us to estimate model parameters in an efficient manner [7]. Third, the dynamics of the manipulation task are modeled according to a linear Gaussian model which depends on state $\mathbf{s}_t$ and on the effects of performing an action $\mathbf{a}_t$ in-state $\mathbf{s}_t$. Fourth, the weighted linear regression coefficients are obtained using least square estimates, and the weighted logistic regression coefficients are obtained using the gradient descent algorithm. Finally tuning the learning rate of the gradient descent is done by adapting the learning rate at each iteration.

The probability of current phase depends on the current effect of performing an action and on the previous phase $p(\rho_t|\mathbf{a}_t, \rho_{t-1})$. The first phase depends on the first effect of action only $p(\rho_1|a_1)$. The dependency of the current phase on the previous is the most important of the model because the model can model hysteresis effects and transient state information which is done with the help of the force data.

In this thesis, we propose a time sequence model that extends the standard HMM to a non-homogenous i.e. time varying scenario. Instead of assuming transition probability at a different time constant, the transition between the hidden phases is modeled by probabilistic classifiers. The time-varying transition matrix $P^{(t)} = [P_{ij}^{(t)}]$ $(p(\rho_t = j|\mathbf{a}_t, \rho_{t-1} = i))$ of the hidden phases is modeled as function of the observed effect performing an action $\mathbf{a}_t$ in-state $\mathbf{s}_t$ at time t. The multi-class logistic regression

(soft-max regression) is used to link the observed effect of performing an action $\mathbf{a}_t$ and the entries of the phase transition matrix $P^{(t)}$

$$p(\rho_t|\mathbf{a}_t, \rho_{t-1}) = \frac{\exp(w_{ij}\phi(\mathbf{a}_t))}{\sum_k \exp(w_{ik}\phi(\mathbf{a}_t))} \tag{29}$$

where $w_{ij} \in \mathbb{R}^{d+1}$ is a weight vector for transitioning from $\rho = i$ to $\rho = j$, and $\phi(\mathbf{a}_t)$ is a function mapping the effect of action to a $d+1$ dimensional feature vector. In our case, we use force as a feature vector.

The initial phase distribution only depends on the first effect of performing an action and is given by

$$p(\rho_1|a_1) = \frac{\exp(w_{0j}\phi(a_1))}{\sum_k \exp(w_{0k}\phi(a_1))} \tag{30}$$

where w_{0j} is the weight vector for each phase.

In this thesis work the probability of effect of performing an action $p(\mathbf{a}_t)$ at time t is assumed to be equal to 1 because learning phases of the model are based on the exploratory actions [7].

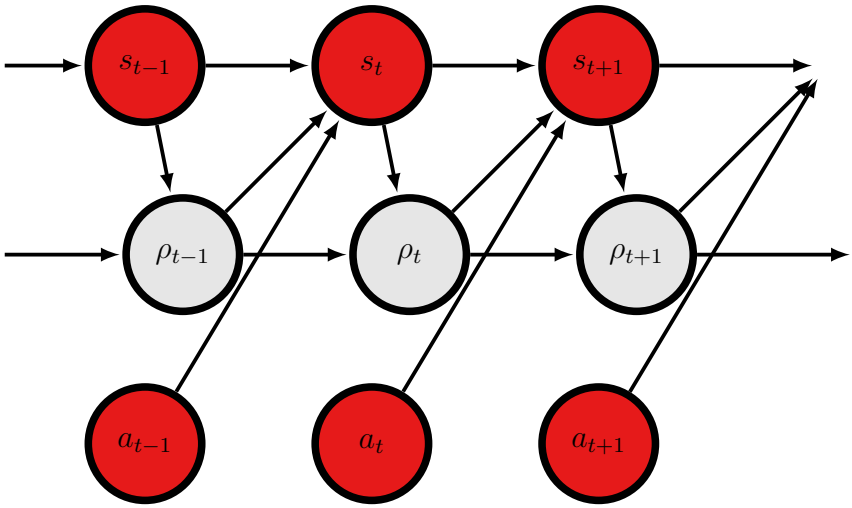


Figure 4: Observed state based transitions auto-regressive HMM (STAR) model [7]

The graphical model of the system is given in Figure 5, where the red filled nodes represent the observed variables i.e. the states $\mathbf{s}_t$ and the effect of the action $\mathbf{a}_t$ and the white filled represent the hidden phases ρ_t. In order to spot the modification of the model description with respect to [7], the model defined by [7] is also shown in Figure 4.

Given the description of the model, the probability of observing a sequence of T samples of states $s_{1:T} = \{s_1, \cdots, s_T\}$, actions $a_{1:T} = \{a_1, \cdots, a_T\}$, phases $\rho_{1:N} = \{\rho_1, \cdots, \rho_T\}$ and the next state $s_{2:T} = \{s_1, \cdots, s_{T+1}\}$ is given by

$$p(s_{1:T+1}, a_{1:T}, \rho_{1:T}) = p(s_1)p(\rho_1|a_1)\prod_{t=1}^{T} p(s_{t+1}|\mathbf{s}_t, \mathbf{a}_t, \rho_t)p(\mathbf{a}_t)\prod_{t=2}^{T} p(\rho_t|\mathbf{a}_t, \rho_{t-1}) \tag{31}$$

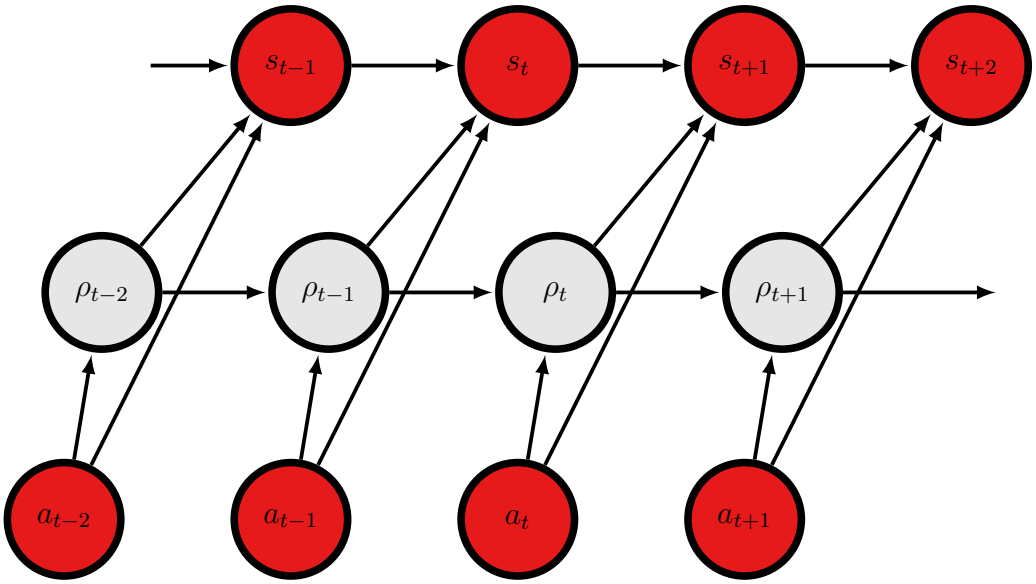


Figure 5: Observed effect of action based transitions auto-regressive HMM (ETAR) model

As seen from Figure 5, the transition between phases depends on the previous phase and the effect of performing an action $\mathbf{a}_t$. The hidden states also satisfy the Markovian property which allows efficient ways to model parameter estimation.

Our model is different from STAR [7] in the following aspects:

1. The state dynamics are modeled by a linear Gaussian model same as [7], but the dimension of effect action matrix changed to $B_i \in \mathbb{R}^{n\times(m+1)}$ instead of $B_i \in \mathbb{R}^{n\times m}$.

2. uses contact forces of the tool concatenated with scalar one as the feature vector instead of a subset of the state vector (relative position) as defined by [7].

3. due to new definition of the feature vector the dimension of the weight vector is now $w_{ij} \in \mathbb{R}^{d+1}$ instead of $w_{ij} \in \mathbb{R}^d$ (for transitioning from $\rho = i$ to $\rho = j$, and $\phi(\mathbf{s}_t)$) which introduces a bias term to the hidden phase transition probability.

4. our model estimation includes estimation of parameters from single and multiple demonstrations with different starting positions.

4.2 Model Selection by AIC and BIC

For HMM with N states, the model order N is directly related with the improvement of the fit of the model. However, an improvement in fit comes with a cost of model parameters quadratic increment. Such improvement in fit due to increasing model number has to be traded off against an increasing number of parameters. A criterion for model selection that solves the trade off between the fit and the number of model parameters is therefore needed.

The goal of the model selection is to identify the model which fits best. The number of hidden state can be selected either with Akaike information criterion (AIC) or Bayesian information criterion (BIC) [50].

Due to large number of parameters, an N -state model will always represent a better fit than an $(N-1)$ state model. Both model selection criterion are used to see if the improvement in the fit was great enough to indicate that the N -state model captures more heterogeneity in data than the $(N-1)$ state model [50]- [55]. Both AIC and BIC, try to find a model that optimally balances the trade off between the model fit and number of model parameters (complexity).

The model selection criterion based on AIC is given by [51]:

$$AIC = -2\log(L) + 2p \tag{32}$$

where $\log(L)$ is the log-likelihood that measures the fit of the model and decreases with increasing model order. The parameter p refers to the number of parameters with in the model and is called a penalty term. The penalty term has direct relationship with model order.

The Bayesian approach another method for estimating model order and differs from AIC in the penalty term [52], [53]. For normally distributed emission probability HMM with N states, there are $N^2 + 2N - 1$ free parameters ($N-1$ for the initial distribution, $N(N-1)$ for the other)

$$BIC = -2\log(L) + p\log(T) \tag{33}$$

where $\log(L)$ and p are as for AIC, and T is the number of observations.

Model selection is based on the AIC/BIC value computed for different orders of the HMM model. The best fit of the model is the model with the lowest value. And because of BIC selects models with fewer parameters compared to AIC, we choose BIC as our model section criteria.

4.3 Model Parameter Estimation

From the model parameter definition of Section 4.1, the system is described by four model parameters, $\theta = \{\mathbf{w}, A, B, \Sigma\}$. These four parameters are the weight vector $\mathbf{w}$ corresponding to the hidden phase, matrices A and Σ corresponding to the observed state and matrix B corresponding to the effect of performing an action.

To learn the parameter of the model, demonstrations are recorded for kinesthetic teaching of a robot a compliant motion based task which includes three phases. Using the recorded position and force data, the EM algorithm is used to estimate the parameters of the model. In the modeling of the system, the observed positions are considered as states and forces as feature vectors of the phase transition probability.

The observed distribution of the next state for each phase of the manipulation task is modeled according to linear Gaussian model

$$s_{t+1} \sim \mathcal{N}(A_{\rho_t}\mathbf{s}_t + B_{\rho_t}\mathbf{a}_t, \Sigma_{\rho_t}) \tag{34}$$

which can equivalently be expressed as

$$s_{t+1} = A_{\rho_t}\mathbf{s}_t + B_{\rho_t}\mathbf{a}_t + v_{\rho_t} \tag{35}$$

where A_{ρ_t} and B_{ρ_t} are state and effect of performing an action matrix respectively, relating the next states linearly to the current states and $\mathbf{v}_{\rho_t}$ Gaussian noise with mean zero and covariance Σ_{ρ_t}. The Gaussian noise $\mathbf{v}_{\rho_t}$ models the uncertainty introduced by the state transition. It has the same dimension as the state vector.

Equation 35 $p(s_{t+1}|\mathbf{s}_t, \mathbf{a}_t, \rho_t)$ can be calculated using the multivariate normal distribution where the mean is given by $\mu_{\rho_t} = A_{\rho_t}\mathbf{s}_t + B_{\rho_t}\mathbf{a}_t$ and the covariance Σ_{ρ_t}

$$p(s_{t+1}|\mathbf{s}_t, \mathbf{a}_t, \rho_t) = \frac{1}{|(2\pi\Sigma_{\rho_t})|^{-\frac{1}{2}}} e^{\left(-\frac{1}{2}\left(s_{t+1}-\mu_{\rho_t}\right)^T \Sigma_{\rho_t}^{-1}\left(s_{t+1}-\mu_{\rho_t}\right)\right)} \tag{36}$$

The EM algorithm finds the maximum likelihood estimates by beginning with an initial guess of the model parameters. It iterates between the E-step and M-step. The algorithm adjusts the parameters of the HMM given a set of observations, using the maximum likelihood criteria. That is it involves the re-estimate of the initial model parameters $\theta = (\mathbf{w}, A, B, \Sigma)$ in order to find the parameters that maximizes conditional probability of observation $p(\mathbf{z}|\theta)$.

4.3.1 Expectation Step (E-step)

During the E-step of the EM algorithm, the marginal likelihood $p(\rho_t = j|a_{1:N})$ and the joint distribution $p(\rho_t = i, \rho_t = j|a_{1:N})$ of the hidden phases given the observed sequence of variables are computed by assuming the model parameters are fixed. During the E-step the marginal likelihood and the joint distribution variables are efficiently evaluated using the forward-backward message passing approach.

To improve the clarity of the methodology we assume a combined observation variable $z_t = \{\mathbf{s}_t, \mathbf{a}_t\}$ for the observed variable states $\mathbf{s}_t$ and effect of the action $\mathbf{a}_t$ as [7]. The joint observation conditional probability is therefore described by $p(z_{t+1}) = p(s_{t+1}|\rho_t, \mathbf{s}_t, \mathbf{a}_t)p(a_{t+1})$ and similarly the phase transition can be described in terms of the joint observation variable as $p(\rho_t|\rho_{t-1}, z_t) = p(\rho_t|\rho_{t-1}, \mathbf{a}_t)$ because as seen from graph description of the model the phase transition is independent of the state $\mathbf{s}_t$.

The forward message per phase which is the joint probability of observing all given observed data up to time t and the value of ρ_t is defined by [7]

$$\alpha_j(t) = p(z_{1:t+1}, \rho_t = j) \tag{37}$$

The first forward message is initialized according to

$$\alpha_j(1) = p(z_2|\rho_1, z_1)p(\rho_1 = j|z_1)p(z_1) \tag{38}$$

and the remaining messages are computed recursively (as the derivation shown in Appendix-A) based on

$$\alpha_j(t) = p(z_{t+1}|\rho_t = j, z_t)\sum_i \alpha_j(t-1)p(\rho_t = j|\rho_{t-1} = i) \tag{39}$$

The backward message is the conditional probability of all future data from $t = N$ given the current phase and next observable data is defined by [7]

$$\beta_j(t) = p(z_{t+1:N}|\rho_t = j, z_{t+1}) \tag{40}$$

Similarly the recursive computation of the backward variable as derived in Appendix-A is given by

$$\beta_j(t) = \sum_i \beta(\rho_{t+1} = i)p(z_{t+1}|\rho_t = i, z_t)p(\rho_{t+1} = i|\rho_t = j) \tag{41}$$

The initialization of the backward message is done at time $t = N$ with

$$\beta_j(N) = 1 \tag{42}$$

The backward recursive computation is done to find the previous messages using

$$\beta_j(t-1) = \sum_i p(\rho_t = i|\rho_{t-1}, z_t)p(z - t + 1|\rho_t = i, z_t)\beta_i(t) \tag{43}$$

Given the forward and backward messages, the intermediate variables i.e. the marginal likelihood $\gamma_t(j) = p(\rho_t = j|z_{1:N+1})$ and the joint distribution $\zeta_t(i, j) = p(\rho_t = i, \rho_t = j|z_{1:N+1})$ can be easily computed. The marginal likelihood $p(\rho_t = j|z_{1:N+1})$ of the phase is given by [7]

$$\gamma_t(j) = p(\rho_t = j|z_{1:N+1}) = \frac{p(\rho_t = j, z_{1:N+1})}{p(z_{1:N+1})} = \frac{\alpha_j(t)\beta_j(t)}{\sum_i \alpha_i(t)\beta_i(t)} \tag{44}$$

Similarly, the joint distribution $\zeta_t(i, j) = p(\rho_t = i, \rho_t = j|z_{1:N+1})$ is calculated using

$$\zeta_t(i, j) = \frac{\alpha_t(t)p(\rho_{t+1} = j|\rho_t = i, z_{t+1})p(z_{t+2}|\rho_{t+1} = j, z_{t+1})\beta_j(t+1)}{p(z_{1:N+1})} \tag{45}$$

4.3.2 Maximization Step (M-step)

During the M-step of the EM algorithm, the parameters of the HMM model are updated by finding the maximum of expected log-likelihood of the joint observed and hidden variables distribution [7].

$$\theta_{new} = \arg\max_\theta \sum_\rho p(\rho_{1:N}|z_{1:N+1}, \theta_{old}) \ln p(p_{1:N}, z_{1:N+1}|\theta) \tag{46}$$

where the summation is over all sequences of phases ρ, and the conditional distributions $p(\rho_{1:N}|z_{1:N+1}, \theta_{old})$ are computed during previous iteration using the un updated model parameters θ_{old}. Using the same simplification criteria as [7] the maximization problem is given by

$$\begin{aligned}
\theta_{new} &= \arg\max_\theta \sum_\rho p(\rho_{1:N}|z_{1:N+1}, \theta_{old}) \ln p(z_1) \\
&+ \sum_\rho p(\rho_{1:N}|z_{1:N+1}, \theta_{old}) \ln p(\rho_1|z_1) \\
&+ \sum_{t=1}^{N} \sum_\rho p(\rho_{1:N}|z_{1:N+1}, \theta_{old}) \ln p(z_{t+1}|\rho_t, z_t) \\
&+ \sum_{t=1}^{N} \sum_\rho p(\rho_{1:N}|z_{1:N+1}, \theta_{old}) \ln p(\rho_t|\rho_{t-1}, z_t)
\end{aligned} \tag{47}$$

By using the same techniques as [7]

$$\begin{aligned}\theta_{new} &= \arg\max_{\theta} \sum_{\rho_1} p(\rho_1|z_{1:N+1}, \theta_{old}) \ln p(\rho_1|z_1) \\ &+ \sum_{t=1}^{N} \sum_{\rho_t} p(\rho_t|z_{1:N+1}, \theta_{old}) \ln p(z_{t+1}|\rho_t, z_t) \\ &+ \sum_{t=1}^{N} \sum_{\rho_{t-1:t}} p(\rho_{t-1}, \rho_t|z_{1:N+1}, \theta_{old}) \ln p(\rho_t|\rho_{t-1}, z_t)\end{aligned} \tag{48}$$

Using the marginal likelihood $p(\rho_t = j|z_{1:N+1}, \theta_{old})$ and the joint distribution $p(\rho_{t-1} = i, \rho_t = j|z_{1:N+1}, \theta_{old})$ of the E-step, the update of model parameters is achieved using the weighted linear regression for parameters related to the state dynamics i.e A_j and B_j and using weighted logistic regression for parameter of the hidden phase transition $\mathbf{w}$.

As defined in Section 4.3.2, the dynamics of the state is defined by

$$s_{t+1} \approx A_j \mathbf{s}_t + B_j \mathbf{a}_t = \begin{bmatrix} A_j & B_j \end{bmatrix} \begin{bmatrix} \mathbf{s}_t \\ \mathbf{a}_t \end{bmatrix} \tag{49}$$

in matrix form it is the product of the row wise concatenation of the model matrices and column wise concatenation of state and effect of performing an action

$$Y = \begin{bmatrix} A_j & B_j \end{bmatrix} X \tag{50}$$

where the additional variable X is the column-wise concatenated variables of the state $\mathbf{s}_t$ and the effect of the action $\mathbf{a}_t$

$$X = \begin{bmatrix} s_1 & \cdots & s_N \\ a_1 & \cdots & a_N \end{bmatrix} \tag{51}$$

and Y is the next state represented by its column

$$Y = \begin{bmatrix} s_2 & \cdots & s_{N+1} \end{bmatrix} \tag{52}$$

Let the sum of weight errors be defined by W_s

$$W_s = \sum_{t=1}^{N} w_{t,j} e_{t,j}^2 \tag{53}$$

its equivalent representation in matrix form is

$$W_s = E_j W_j E_j^T = \left(Y - \begin{bmatrix} A_j & B_j \end{bmatrix} X\right) W_j \left(Y - \begin{bmatrix} A_j & B_j \end{bmatrix} X\right)^T \tag{54}$$

the expanded form of the above equation is

$$\begin{aligned} W_s = \; & Y^T W_j Y - Y^T W_j \begin{bmatrix} A_j & B_j \end{bmatrix} X - X^T \begin{bmatrix} A_j & B_j \end{bmatrix}^T W_j Y + \\ & X^T \begin{bmatrix} A_j & B_j \end{bmatrix}^T W_j \begin{bmatrix} A_j & B_j \end{bmatrix} X \end{aligned} \tag{55}$$

which is simplified to

$$W_s = Y^T W_j Y - 2X^T \begin{bmatrix} A_j & B_j \end{bmatrix}^T W_j Y + X^T \begin{bmatrix} A_j & B_j \end{bmatrix}^T W_j \begin{bmatrix} A_j & B_j \end{bmatrix} X \quad (56)$$

We wish to find A_j and B_j which minimizes the weighted sum error W_s. Therefore the parameter estimate A_j and B_j can be derived by taking the derivative of sum of weighted errors on the parameters A_j and B_j as follows [54]

$$\frac{\partial}{\partial A_j, B_j} W_s = 0 - 2X^T W_j Y + 2X^T W_j X \begin{bmatrix} A_j & B_j \end{bmatrix}^T = 0 \quad (57)$$

The new estimates of the parameter matrices are then given by

$$\begin{bmatrix} A_j & B_j \end{bmatrix}^T = \begin{bmatrix} A_j \\ B_j \end{bmatrix} = \left(X W_j X^T\right)^{-1} X^T W_j Y \quad (58)$$

where the T indicates the transposes of the matrices, and W_j is a diagonal matrix, where the t^{th} diagonal element is given by $[W]_{tt} = p(\rho_t = j | a_{1:N+1}, \theta_{old})$ as in [7].

Consider the case when the observed state $\mathbf{s}_t$ is i.i.d. linear Gaussian with mean vector and covariance matrix depending on the current phase ρ_t:

$$s_{t+1} \sim \mathcal{N}(A_{\rho_t} \mathbf{s}_t + B_{\rho_t} \mathbf{a}_t, \Sigma_{\rho_t}) = \mathcal{N}(\mu_{\rho_t}, \Sigma_{\rho_t}) \quad (59)$$

The p.d.f can be written as [36]

$$p(s_{t+1} | z_{t-1}, \rho_{t-1}, \theta) = \frac{1}{\sqrt{(2\pi)^n}\sqrt{\Sigma_{\rho_t}}} \exp\left(\frac{-(s_{t+1} - \mu_{\rho_t})\Sigma_{\rho_t}^{-1}(s_{t+1} - \mu_{\rho_t})}{2}\right) \quad (60)$$

Differentiating Equation 60 with respect to the mean vector and the covariance matrix gives us

$$\partial \frac{p(s_{t+1} | \mathbf{s}_t, \theta)}{\partial \mu_j} = \begin{cases} \Sigma_j^{-1}(s_{t+1} - \mu_{\rho_t}) & if \quad \rho_t = j \\ 0 & otherwise \end{cases} \quad (61)$$

$$\partial \frac{p(s_{t+1} | \mathbf{s}_t, \theta)}{\partial \Sigma_j^{-1}} = \begin{cases} \frac{1}{2}\Sigma_j^{-1} - \frac{1}{2}(s_{t+1} - \mu_{\rho_t})^T (s_{t+1} - \mu_{\rho_t}) & if \quad \rho_t = j \\ 0 & otherwise \end{cases} \quad (62)$$

Taking the summation of the partial derivative over all the hidden phases as indicated on [17] will result in the new estimate of the covariance matrices

$$\Sigma_j = \frac{\sum_{i=1}^{N} p(\rho_i = j | s_{1:N+1}, \theta_{old})(s_{i+1} - \mu_{ji})^T (s_{i+1} - \mu_{ji})}{\sum_{k=1}^{N} p(\rho_k = j | s_{1:N+1}, \theta_{old})} \quad (63)$$

where $\mu_{ji} = A_j s_i + B_j a_i$ is the expected next state for the reestimate state parameters A_j and B_j.

The probabilistic classifier parameter (weight vector) $\mathbf{w}$ is calculated using weighted logistic regression. Logistic regression is calculated iteratively using gradient descent as in [7]. The gradient of the $j-th$ phase given the $i-th$ phase can be obtained by derivation with respect to the weight vector.

$$G = \nabla p(\rho_t|\mathbf{a}_t, \rho_{t-1}) = \nabla \frac{\exp(w_{ij}\phi(\mathbf{a}_t))}{\sum\limits_k \exp(w_{ik}\phi(\mathbf{a}_t))} \tag{64}$$

$$G = \nabla p(\rho_t|\mathbf{a}_t, \rho_{t-1}) = \frac{\frac{d}{dw_{ij}}\left(e^{w_{ij}\phi(\mathbf{a}_t)}\right)\sum\limits_k e^{w_{ik}\phi(\mathbf{s}_t)} - e^{w_{ij}\phi(\mathbf{a}_t)}\frac{d}{dw_{ij}}\left(\sum\limits_k e^{w_{ik}\phi(\mathbf{a}_t)}\right)}{\left(\sum\limits_k e^{w_{ik}\phi(\mathbf{a}_t)}\right)^2} \tag{65}$$

after simplification of Equation 65 as in [55], the gradient is given by

$$G = \nabla p(\rho_t|\mathbf{a}_t, \rho_{t-1}) = \phi(\mathbf{a}_t)\left(\delta_{ij} - p(\rho_t|\mathbf{a}_t, \rho_{t-1})\right) \tag{66}$$

where δ_{ij} in Equation 66, represents the joint distribution from E-step. For simplified computation of the gradient, for transition from $\rho_{t-1} = i$ three matrices are defined as [7]

1. F is a feature matrix with the columns containing the sampled effect of performing an action $\phi(\mathbf{a}_t)$.
2. The matrix L obtained from E step and it is defined by $[L]_{tj} = p(\rho_{t-1} = i, \rho_t = j|z_{1:N+1}, \theta_{old})$. The L matrix contains the joint hidden phase distribution.
3. The matrix P has the same dimension as the L matrix, and it contains the hidden phase transitions $p(\rho_t = j|\mathbf{a}_t, \rho_{t-1} = i)$ for a given weights $\mathbf{w}$.

The gradient of the hidden phase transition with respect to $\mathbf{w}$ is equivalent representation of Equation 66

$$G = F(P - L) \tag{67}$$

In this iterative process between the E-step and M-step, the HMM utilizes two mathematical principles to estimate model parameters as shown in Figure 6. The first is the E-step which is based on the forward-backward algorithm. For each observation data points, the algorithm evaluates the forward and backward variables, α and β. The forward algorithm is used to compute the forward variable and the backward algorithm for the backward variable.

By using, the forward and backward variables, intermediate variables γ and ζ are obtained. By using those variables model parameters update is done during M-step. The M-step of the EM algorithm is based weighted linear regression and weighted logistic regression. The update of the model parameters is done based on the intermediate variables obtained from the E-step. The algorithm continues iteration either until the maximum number of iterations or until the convergence.

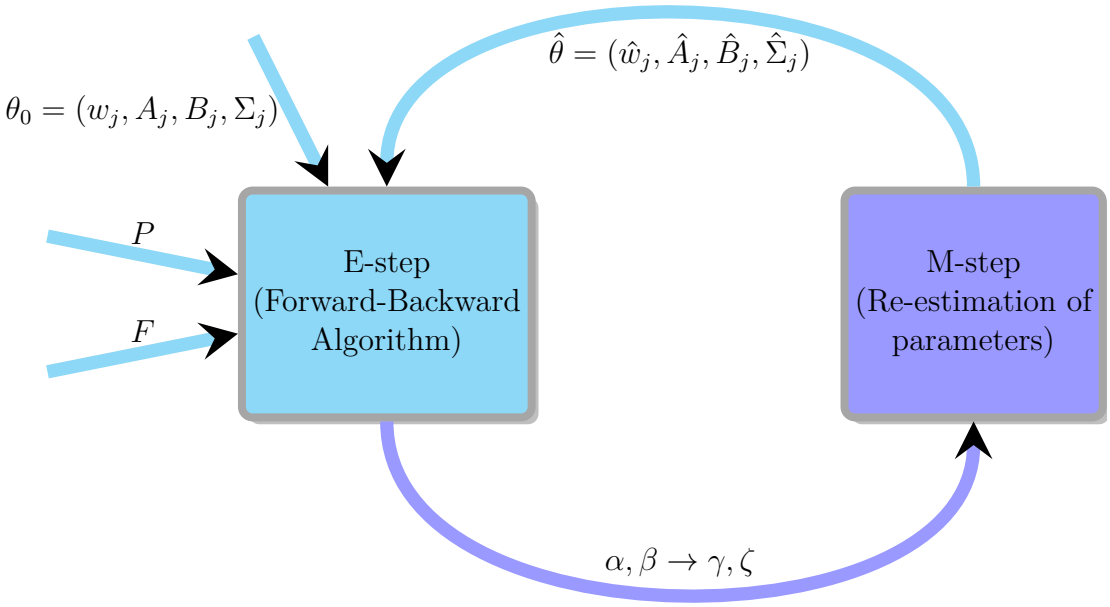


Figure 6: Block diagram representation of the EM algorithm

4.4 Optimization Method: Tuning Learning Rate

As mentioned in Section 4.5, the logistic regression of the hidden phase distribution is optimized using a gradient descent algorithm. Gradient descent exploits first-order local information in the gradient to iteratively approach the point at which the multi-class logistic regression distribution achieves its minimum value. The weight vector at each iteration can be expressed as a function of the gradient as follow

$$\mathbf{w}_{k+1} = \mathbf{w}_k - \lambda G = \mathbf{w}_k - \lambda \left(F(P - L)\right) \tag{68}$$

where λ is step size of the gradient descent algorithm which is commonly known as learning rate.

One of the problems in the updating the weight vector i.e. choosing the right value of learning rate. The step size can be constant or variable. In practice, determining a fixed step size that is adequate for a particular function can be challenging because choosing a step size which is too small results in slow convergence and on the other side selecting a step size which is too large may cause repeatedly overshoot the minimum and eventually diverge.

Automatic adaptation of the step size as the iterations progress is a solution for the above problem. One of the solutions is to search for the minimum of the function along the direction of the gradient which is commonly known as line search. To minimize the cost of computation a modified version of the line search called backtracking line search can be used.

The backtracking line search starts at large value of λ and decreases it until the logistic function is below $f(\mathbf{w}) - \frac{1}{2}||\Delta f(\mathbf{w})||_2^2$, a condition known as Armijo rule [56].

Gradient descent can be included in the M-step of EM algorithm and run for a certain number of iterations or until convergence criterion, before moving the E-step which might depend on computational constraints, or until a stopping criterion is met. The possible termination rule for the logistic regression function $f(\mathbf{w})$ can be

1. $\Delta f(\mathbf{w}^{k+1}) = 0$
2. $\left| f(\mathbf{w}^{k+1}) - f(\mathbf{w}^{k}) \right| < \epsilon$
3. $||\mathbf{w}^{k+1} - \mathbf{w}^{k}|| < \epsilon$

The flow chart of the EM algorithm including the internal loop gradient descent update of the weight vector is shown in Figure 7.

4.5 Model Parameter Estimation from Multiple Demonstrations

For estimating model parameters from multiple demonstrations an EM algorithm which similar EM algorithm from a single demonstration data is used.

During the E-step the forward and backward messages are separately computed for each demonstration data's with same procedure as described in Section 3.4.1. From the forward and backward messages the intermediate variables marginal likelihood and the joint probabilities of the hidden phases is also derived from the forward-backward algorithm variables.

During the maximization step a combined marginal likelihood variable γ is defined which is a concatenation of the marginal likelihood of multiple demonstrations $\gamma_1, \gamma_2, ...\gamma_D$.

$$\gamma_D = \begin{bmatrix} \gamma_1 & \gamma_2 & \cdots & \gamma_D \end{bmatrix} \tag{69}$$

Similar to the single demonstration case a new definition of the matrix X and Y is assigned in order to estimate the state and effect of action matrices using weighted linear regression. Each column of matrix X consists of the concatenated state s and action a of the sample for both demonstration combined and matrix Y, has $N_1 + N_2 + \cdots N_D$ columns corresponding to the a sampled next state of multiple demonstrations.

$$X = \begin{bmatrix} s_1^{(1)} & \cdots & s_{N_1}^{(1)} & s_1^{(2)} & \cdots & s_{N_2}^{(2)} & \cdots & s_1^{(D)} & \cdots & s_{N_D}^{(D)} \\ a_1^{(1)} & \cdots & a_{N_1}^{(1)} & a_1^{(2)} & \cdots & a_{N_2}^{(2)} & \cdots & a_1^{(D)} & \cdots & a_{N_D}^{(D)} \end{bmatrix} \tag{70}$$

where the superscript $(1), (2), \cdots, (D)$ is used to indicate the data is from demonstration $1, 2, \cdots D$ respectively and similarly Y is defined as

$$Y = \begin{bmatrix} s_1^{(1)} & \cdots & s_{N_1}^{(1)} & s_1^{(2)} & \cdots & s_{N_2}^{(2)} & \cdots & s_1^{(D)} & \cdots & s_{N_D}^{(D)} \end{bmatrix} \tag{71}$$

The new estimates of the parameter matrix from the combined demonstrations are then given by

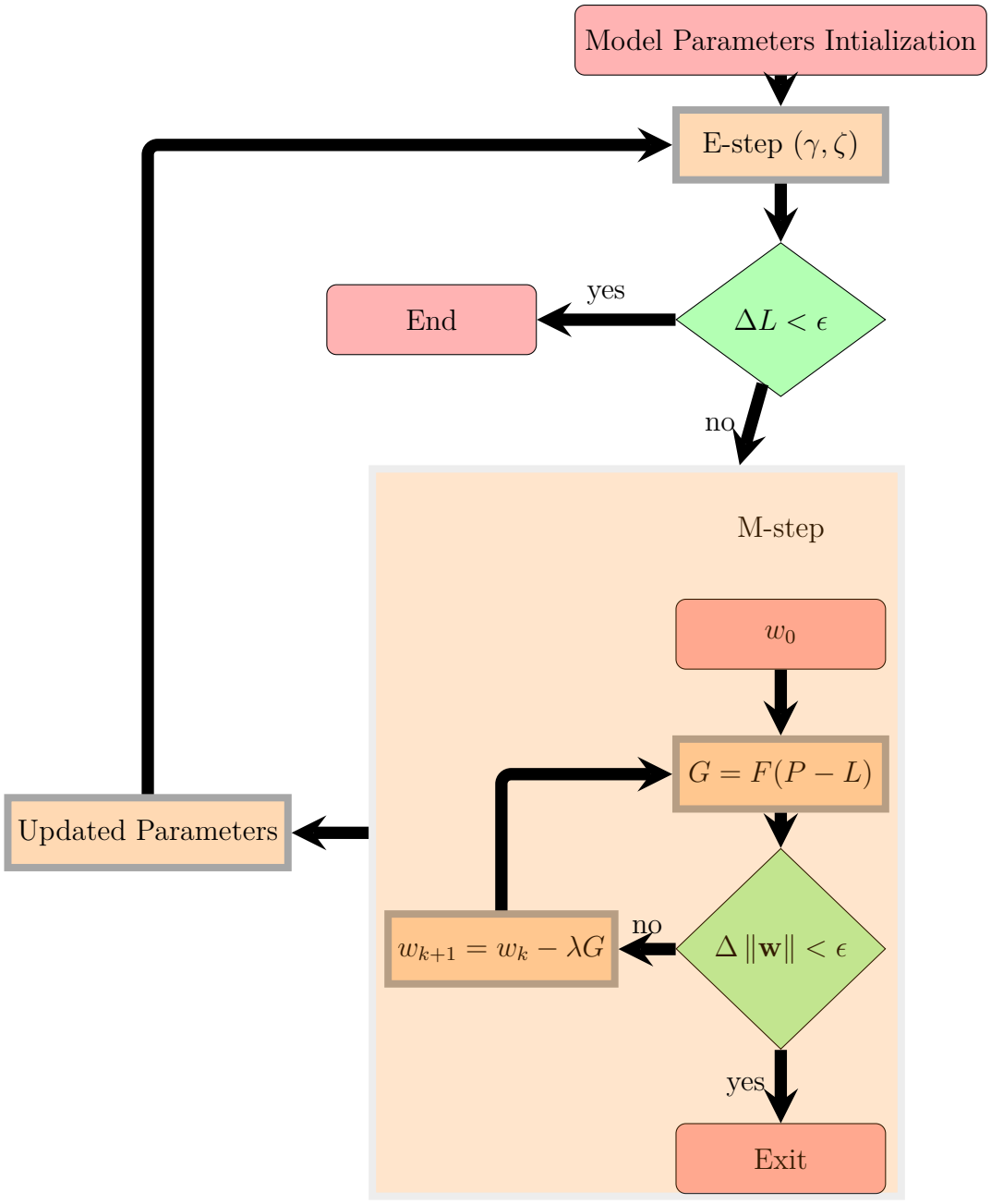


Figure 7: Simplified Flow chart EM algorithm with logistic regression

$$\begin{bmatrix} A_j & B_j \end{bmatrix} = YW_jX^T\left(XW_jX^T\right) \tag{72}$$

where W_j is the modified combined diagonal matrix corresponding to γ_D, where the t^{th} diagonal element is given by $[W]_{tt} = p(\rho_t = j|a_{1:N+1}, \theta_{old})$.

The gradient computation of the logistic regression is also modified as we have separate values of F, P and L for each demonstration. Therefore the combined gradient G_{12} is calculated by

$$G_D = F^{(1)}(P^{(1)} - L^{(1)}) + F^{(2)}(P^{(2)} - L^{(2)}) + \cdots + F^{(D)}(P^{(D)} - L^{(D)}) \tag{73}$$

where the variables $F^{(D)}, P^{(D)}$ and $L^{(D)}$ have the same definition as F, P and L corresponding to each demonstration. Then the update of the weight vector is done using

$$\begin{aligned} \mathbf{w}_{k+1} &= \mathbf{w}_k - \lambda G_D \\ &= \mathbf{w}_k - \lambda \left(F^{(1)}(P^{(1)} - L^{(1)}) + F^{(2)}(P^{(2)} - L^{(2)}) + \cdots + F^{(D)}(P^{(D)} - L^{(D)}) \right) \end{aligned} \tag{74}$$

In each iteration the smoothed probabilities such as marginal likelihood and joint distribution of the phase are calculated and solve the new parameters of the model $\theta_{new} = (\mathbf{w}, A, B, \Sigma)$ as a function of the previous estimation θ_{old}.

For the expected next state given the updated A and B matrices for each demonstrations $\mu_{ji}^{(1)} = A_j s_i^{(1)} + B_j a_i^{(1)}, \mu_{ji}^{(2)} = A_j s_i^{(2)} + B_j a_i^{(2)}, \cdots, \mu_{ji}^{(D)} = A_j s_i^{(D)} + B_j a_i^{(D)}$, we define a new variable Δs, which is equal to

$$\Delta s = \begin{bmatrix} \left(s_{t+1}^{(1)} - \mu_{ji}^{(1)}\right) & \left(s_{t+1}^{(2)} - \mu_{ji}^{(2)}\right) & \cdots & \left(s_{t+1}^{(D)} - \mu_{ji}^{(D)}\right) \end{bmatrix} \tag{75}$$

Using the combined Δs, the new estimate of the covariance matrices

$$\Sigma_j = \frac{\sum_{i=1}^{N} p(\rho_i = j | s_{1:N+1}, \theta_{old})(\Delta s)^T (\Delta s)}{\sum_{k=1}^{N} p(\rho_k = j | s_{1:N+1}, \theta_{old})} \tag{76}$$

As an example the modified simplified flow chart of the EM algorithm for combining two demonstration is shown in Figure 8

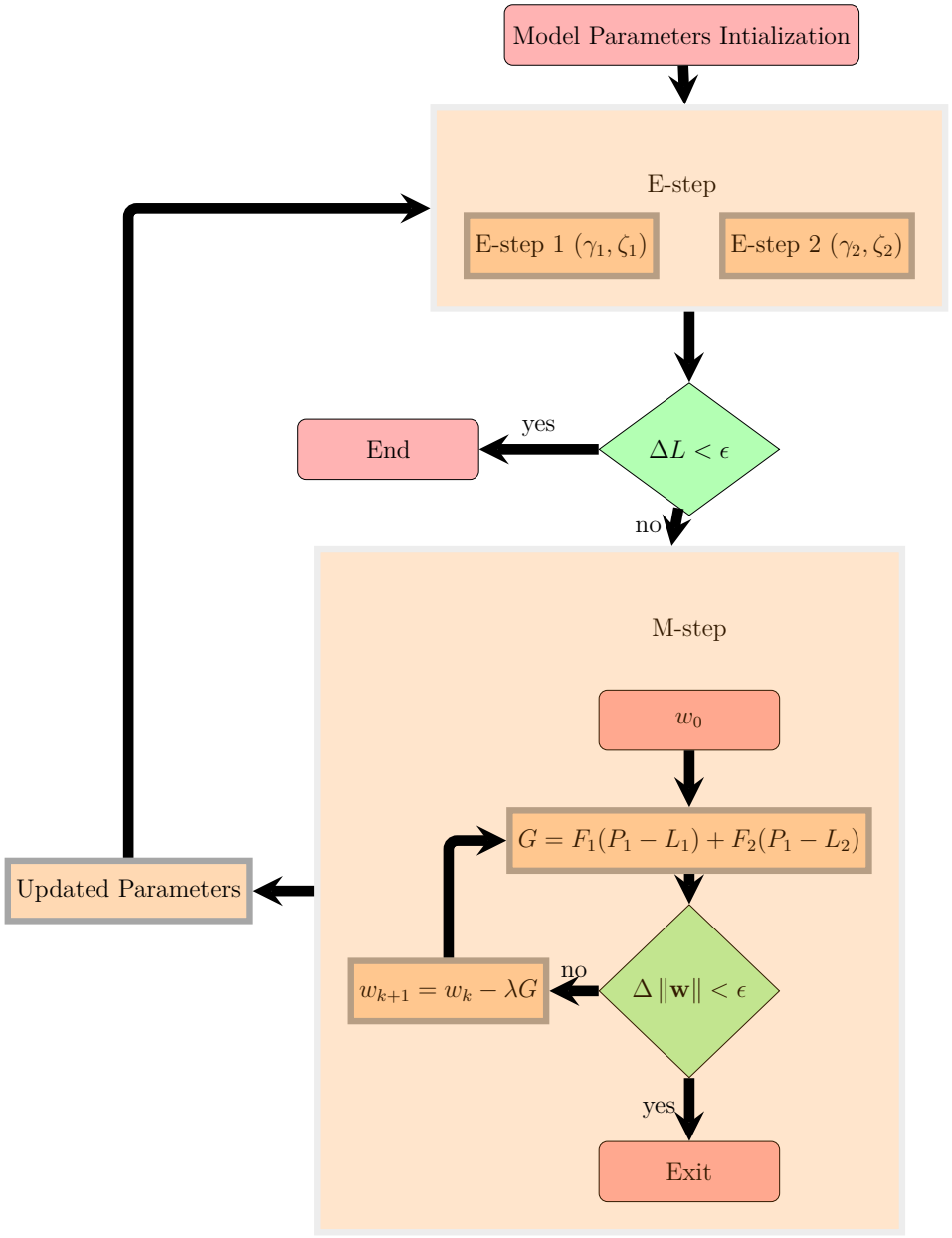


Figure 8: Simplified Flow chart of EM algorithm with logistic regression for two demonstrations

5 Hardware and Software Architecture

The implementation and testing of the approach developed in this thesis needs an integrated hardware and software infrastructure to carry out the experiments. This section describes the HW/SW architecture required and the detailed implementation procedure followed during the experiment will be discussed

5.1 Software Architecture

The LfD approach involves three phases. First, an operator demonstrates the task and data required for the learning phase is recorded. Then, during the learning phase, the position and force data obtained from the teaching phase are used to segment and estimate the parameters of the model of each segmented manipulation task. The final step is reproduction phase. During this phase, the model learned during the learning phase are used to reproduce the targeted manipulation task. The block diagram representation of the three stages of LfD paradigm presented in Figure 9.

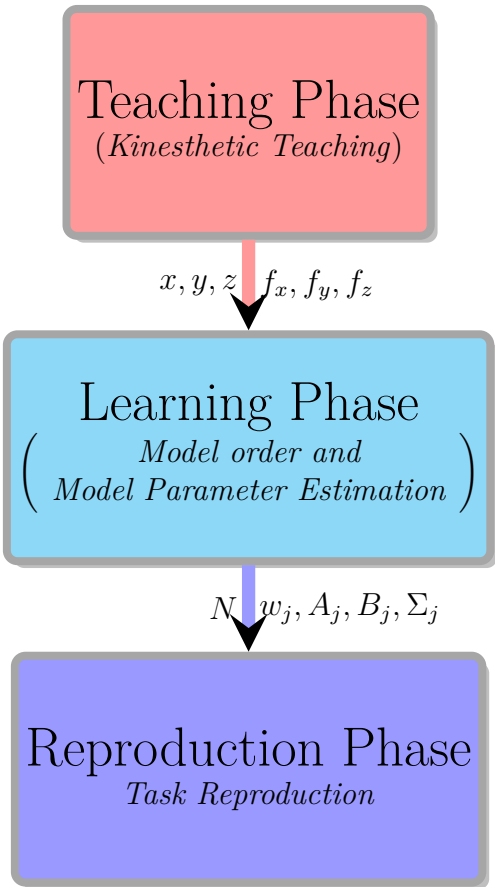


Figure 9: Block diagram representation of the three phases involved in LfD approach

Therefore the software architecture includes the components required for these three phases. All the ROS (Robot Operating System) packages and Orocos (Open RObot Control Software) nodes are written in C++. The existing frame work

includes the Orocos components such as the "FT Senor", "KUKACommanderROS", "KUKACommander" and "FRI Server" for real-time and ROS for all other parts [45]. The learning phase of the HMM is done in Matlab environment. The controller component (Stiffness_phase_controller) that makes use of off-line model parameters estimated is the developed Orocos component.

5.1.1 Teaching Phase

During the teaching phase, a human teacher guides the robot to accomplish the manipulation task (aka kinesthetic teaching). The LfD is carried out using one component from the basic framework, the ROS package Recorder. First, the Recorder asks the user for the trajectory type (i.e. Cartesian or Joint space) and the inclusion of force measurements from the F/T sensor. Based on these choices, the Recorder subscribes to different topics. Then, using the services provided by the KUKACommanderROS, the arm is put in gravity compensation. Therefore, the arm can be easily moved, and the task can be demonstrated in kinesthetic teaching. After reaching the starting position, the measurements published by the FRIServer and the FTSensor are recorded to bag files with a frequency of 100 Hz. After one demonstration of the task, the arm is moved to the starting position. The program allows the teacher to demonstrate several times the task, recording each demonstration in separate bag files.

5.1.2 Learning Phase

During the learning phase, the HMM is trained using EM algorithm and the required parameters are estimated. The training of HMM is based on the data obtained from teaching phase and it is done offline using Matlab as described in Section 4.3.

5.1.3 Reproduction Phase

Assuming the reproduction step starts from phase one, for the first time instant the controller defined for the starting phase is used to initiate the reproduction.

We use the learned parameters for each phase to iteratively predict the current phase based on the previous position i.e. the state and force i.e. the effect of performing an action. Orocos component consists of a controller that make use of the learned HMM model have been developed. The controller is an orocos component which consists of a predefined Cartesian impedance controller parameters as in [6] for each phase of the task.

The next step is to compute the next state prediction using the model parameters estimated and the previous state and effect of action and solve for the difference between the actual value and the predicted value. Using the difference between the actual value and the predicted value and the estimated covariance matrix for each phase, a state transition probability is computed. Finally, using the on-line force and position data, the forward message is calculated based on the phase transition probability and previous forward message and state transition probability. Normalizing the forward message gives us the probability phase measure. The most

likely phase is obtained from the forward message and then based on the entries of the forward variable corresponding to each phase of the controller we defined desired direction of movement, the number of compliant axes and their directions corresponding to the impedance controller for each phase. The impedance controller used for reproduction is a feed back controller defined by [6]

$$F = K(\mathbf{x}^* - \mathbf{x}) + D + \mathbf{f}_{dyn} \tag{77}$$

where $\mathbf{x}^*$ is desired position, $\mathbf{x}$ is current position, K is gain matrix, D is linear damping term and $\mathbf{f}_{dyn}$ is the feed-forward dynamics of the robot. Assuming the direction of compliance learned is represented by a rotation matrix R, the stiffness matrix of the controller is defined by [6]

$$K = RVR^T \tag{78}$$

5.2 Hardware Architecture

The hardware of the experiment includes the following components as shown in Figure 10.

- KUKA Light Weight Robot (LWR4+) (1),
- F/T sensor (2),
- KUKA Robot Controller (KRC) (3) with KUKA Control Panel (KCP) (4) and
- an external computer (5).

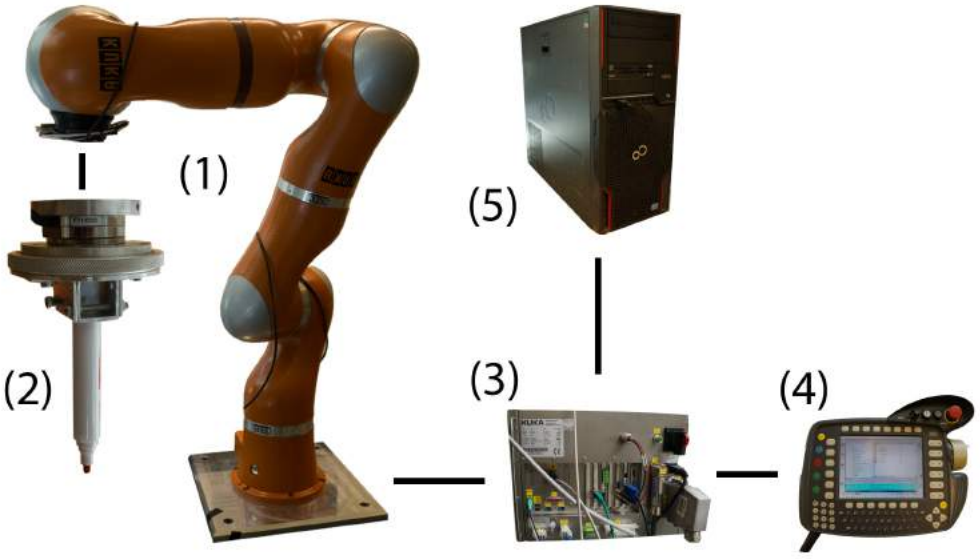


Figure 10: Hardware setup (Source: [42])

The KUKA LWR4+ robot which is a kinetically redundant robot with 7 DOF designed to overcome the limits of industrial robots in unstructured human environments [43]. The robot weighs 15 kg and is able to handle payloads of 7 kg. The two main features offered by the LWR which make for implementation of the thesis experiments are

1. Actively compensate robot weight due to accurate dynamic model and force-torque sensing

2. Compliant motion behavior by setting the parameters of the robot impedance controller (stiffness and damping)

The ATI Multi-Axis Force/Torque Sensor is an external sensor attached to the tool used to measure the contact force and torque. The sensor can measure up to 290 N in x and y directions, 580 N in the z-direction for force and up to 10 Nm in all directions for torque [44].

The LWR is connected to the KRC, a computer that controls the arm and ensures safe operation and controls the robot.

Both the KRC and the F/T sensor are connected to the external computer with dedicated UDP connections over Ethernet. Due to the hard real-time requirements of the communication, the external computer is equipped with two additional network cards and runs a real-time version of Linux [45].

The external computer is an octa-core computer using Linux operating system Ubuntu 14.04 LTE with a Xenomai real-time kernel. The KUKA's Fast Research Interface (FRI) is used to control the robot through the external computer [45].

5.3 Cartesian Impedance Controller

The targeted manipulation task of this thesis or in general assembly task is based on the in-contact motion of the tool with the environment. To use the compliant behavior of the controller a Cartesian impedance controller integrated into the LWR arm is used. The Cartesian impedance controller works in conjunction with the torque controllers of the joints. If this controller is used, the robot behaves like a spring/damper system with stiffness and damping parameters defined along the axes of the chosen frame [48], [6].

The control law implemented can be described as

$$\tau_{cmd} = J^T(k_c(x_{FRI} - x_{msr}) + D(d_c) + F_{FRI}) + f_{dynamics}(q, \dot{q}, \ddot{q}), \tag{79}$$

where J^T is the Jacobian converting forces at the end-effector in the resulting joint torques τ_{cmd}. The control law represents a virtual spring $k_c(x_{FRI} - x_{msr})$. The Cartesian stiffness of the virtual spring k_c, the Cartesian normalized damping parameter d_c, the desired Cartesian position x_{FRI} and the superposed force/torque term F_{FRI} can be dynamically set. The term $f_{dynamics}(q, \dot{q}, \ddot{q})$ takes care of the dynamic model of the robot and compensates for gravity torques, Coriolis and centrifugal forces [45], [49].

If the stiffness parameters k_c are set to zero, the desired position x_{FRI} will not be reached and only the superposed force F_{FRI} will be controlled. The impedance controller will move the arm in the direction of the force to be exerted and provide the requested force if some reaction is found. It is noteworthy to recall that the two force components composing the control law (the virtual spring force $F_k = k_c(x_{FRI} - x_{msr})$ and the superposed force component F_{FRI}) can counteract each other.

Finally, if the stiffness k_c and the superposed force F_{FRI} are set to zero, the robot is controlled in the same way as with *gravity compensation*. The robot can be freely moved while gravitational and frictional torques are computed and then compensated. However, the positioning accuracy on the set point x_{FRI} in the individual Cartesian direction is influenced by the parameters of the impedance controller. In case of low stiffness or small relative displacement $(x_{FRI} - x_{msr})$, it is possible that the spring force F_k is no longer sufficient to overcome static friction [48].

6 Experiments and Results

As mentioned in Section 4, the first research question if model order can be identified. The second question is how precise is the prediction of phase transition from the model learned.

The main objective was to segment phases of a manipulation task and find model parameters that best fit the phases of each task. Then, based on the obtained model parameters, we tried to identify the phases during reproduction and activate an impedance controller defined by its corresponding direction of motion, compliant axes, and their directions for each segmented phase.

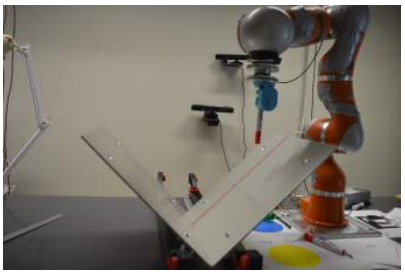
The first experimental setup was a valley consisting of two aluminum plates set on 45 degrees angle on the table as shown in Figure 11 with single starting position (SP). The demonstrated task included three phases (i.e. one unconstrained motion and two constrained motions): non-contact, sliding along one of the aluminum plate set on 45 degrees angle and sliding across the intersection of the two aluminum plates.

Figure 11: Experimental setup for single demonstration

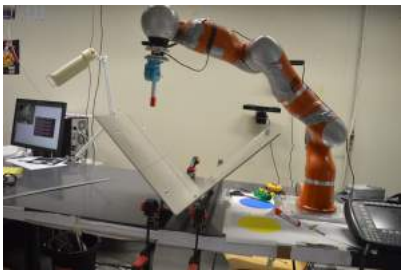
The second setup was similar to the first experimental setup except that it had two different SPs, one from left and the other from the right side of the setup as shown in Figure 12a and 12b respectively. The demonstrated two tasks included three phases one unconstrained motion and two constrained motions. The first demonstrated include non-contact, sliding along the first aluminum plate set at 45 degrees angle and sliding across the intersection of the two aluminum plates with starting position shown in Figure 12a. The second demonstrated task was the same as the second demonstrated task, however it had different starting position as illustrated in figure 12b and the sliding phase is along the second metal plate.

The experiments we conducted included

- Kinesthetically teaching the robot and recording data
- Model order and model parameter estimation from position and force data
- Reproduction of the manipulation task based on the estimated model parameters by choosing a controller that best fits with the estimated phase



(a) Experimental setup from SP 1



(b) Experimental setup from SP 2

Figure 12: Experiment setup for demonstrations from two different SPs

6.1 Model parameter Estimation from Single Demonstration

In the model parameter learning from a single demonstration, six similar demonstrations starting from the same starting position were used. During the kinesthetic teaching of the robot, we recorded the position and the forces of the tool as shown in Figure 13a and Figure 13b respectively. The first phase is represented by decreasing position of z-component while the other components remain constant, the second phase is represented by increasing y-component and decreasing z-component and the last phase is represented by decreasing x-component and uniform other components as shown in Figure 13a. Each phase of the manipulation task is represented with specific feature vector as shown Figure 13b: zero contact force for the first phase, a sudden increase of force in the y and z components and the last phase is represented by a non-zero force along all components.

Then BIC was used for estimating the model number as described in Section 4.2 for all the six separate demonstrations. The estimated model number was consistent as shown in Figure 14 which is equal to $N = 3$.

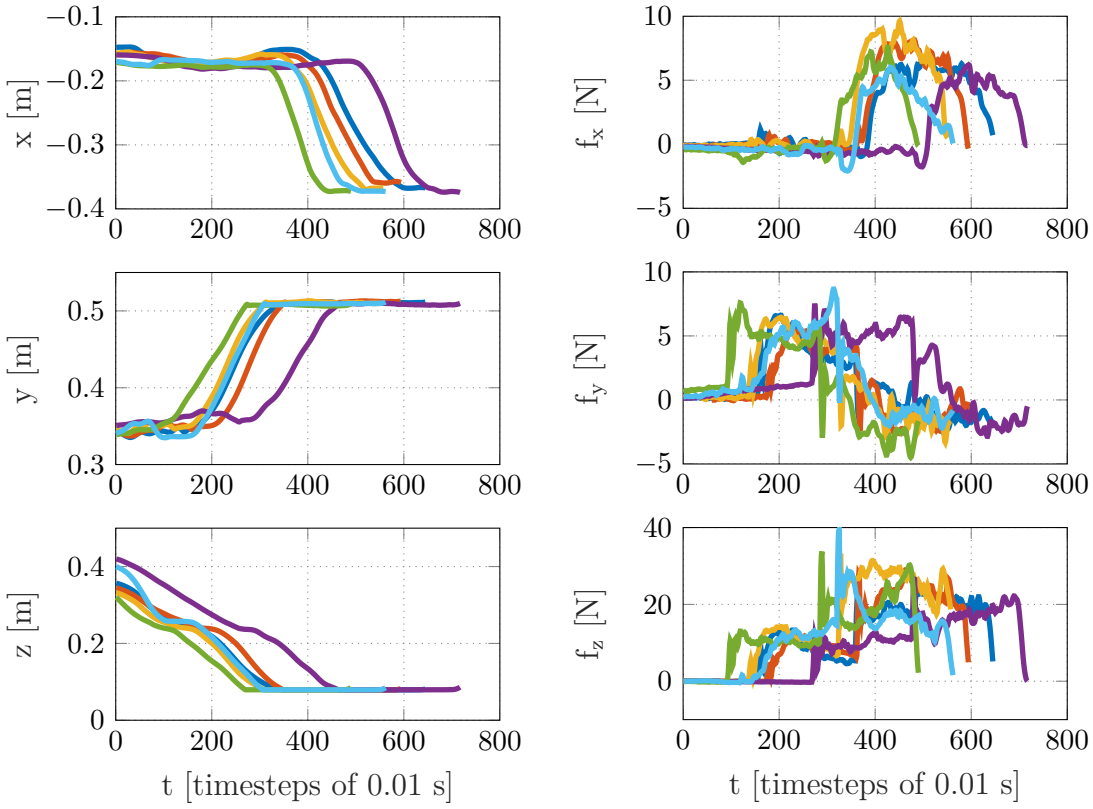
BIC model selection criterion plot of Figure 14 is with respect to a possible number of states N of the HMM. According to BIC, the model with three states ($N = 3$) was the most appropriate model number for the modeled manipulation task. The implemented BIC was also compared with as BIC Matlab function "bic" as shown in the Figure 14. Our implemented BIC was specifically designed for Gaussian emission with defined penalty term that is why the two graphs do not overlap.

Using the estimated model number we ran the EM algorithm to estimate the model parameters. The algorithm was able to predict the phases of the assembly task as shown in the Figure 15a which corresponds to the normalized forward variable of HMM α. The marked 3D plot of the position of the tool corresponding to the estimated sequence of phase Figure 15a is shown in Figure 15b

To validate how well the estimated parameters for each phase can predict the dynamic behavior of the observed state of the model, we used the model parameters of each phase to predict next state.

$$\hat{s}_{t+1} \approx A_j \mathbf{s}_t + B_j \mathbf{a}_t \tag{80}$$

where $j \in [1, 2, 3]$, corresponding to the estimated phase. And we subtract the



(a) position of the tool (b) contact force of the tool

Figure 13: Position and force of the tool for six similar demonstrations (each color is a demonstration)

predicted state from the actual measured to find error of prediction for each phase using

$$\hat{e}_t \approx \hat{s}_t - \mathbf{s}_t \tag{81}$$

The average error bar of the prediction corresponding to the three phases estimated parameters is shown in Figure 16, 17 and 19 respectively. As shown in the Figure 16, the state prediction using the model parameters of the initial phase is able to predict well the first phase as it is represented with lower error until the first vertical line compared with the other two phases.

Using the same principle, the predicted states using the model parameters of the second phase were able to predict the second phase accurately as it is shown in Figure 17 with lower error in the region between the two vertical lines.

From Figure 17, the prediction error of the first phase and the second phase states obtained from the second phase model parameters looks like the same, but they are not. We took a zoomed version of the phase and part phase two to see the difference. As shown in Figure 18, the prediction error of phase two is smaller than phase two which indicates a better prediction of phase two compared other phases.

The prediction of states using the last phase model parameters were also done as

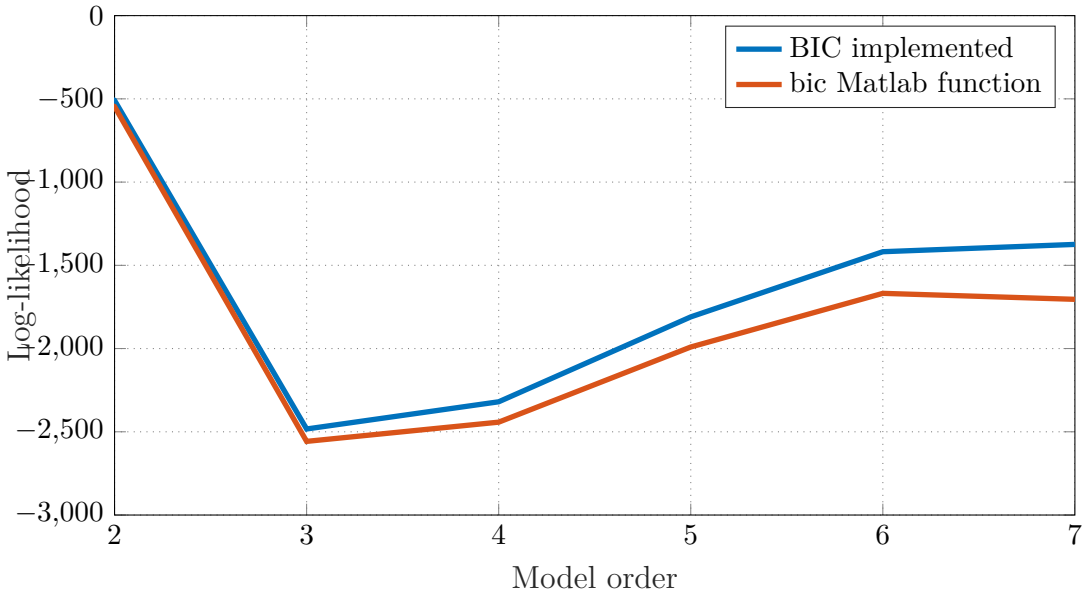
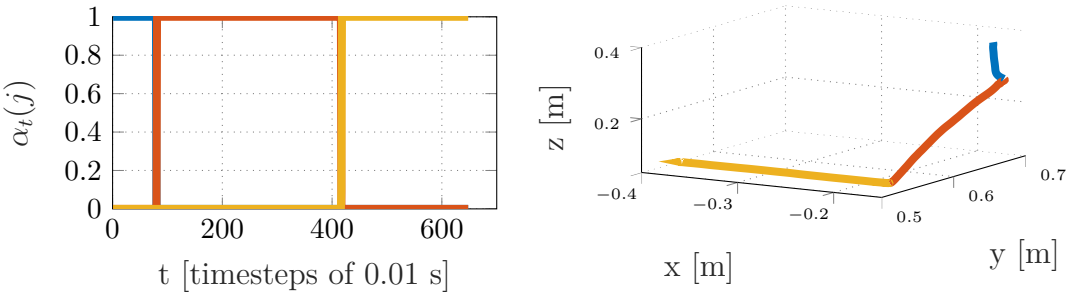


Figure 14: BIC based estimated model number using single demonstrations



(a) Estimated phases (normalized α)

(b) Position of the tool

Figure 15: Estimated phases and the corresponding position of the tool for the target manipulation task

shown in Figure 19. As indicated in the figure, the phase after the second vertical is represented with lower error which indicates the correct prediction of phase 3.

To support the goodness of our prediction of state from the learned model parameters, a qualitative measure of the average error variance was made. Here $E_{\rho_i}(\theta_j)$ represents the average error of phase i (ρ_i) due to the prediction of state using model parameter θ_j. As shown in Figure 20, state predicted using the model parameters of the first phase has lower error compared to other phase predictions using the same model parameters. All yellow colors bar are average error variance related to phase 1 and all magenta, and cyan colors are related to phase two and three respectively. The pair of three colors represents average prediction error variance of a phase, starting from phase one and then phase two and final phase three. Comparing the first three colored bar, the bar with label $E_{\rho_1}(\theta_1)$ has lower average error variance which indicates a good prediction of phase one ρ_1 using model parameters of phase

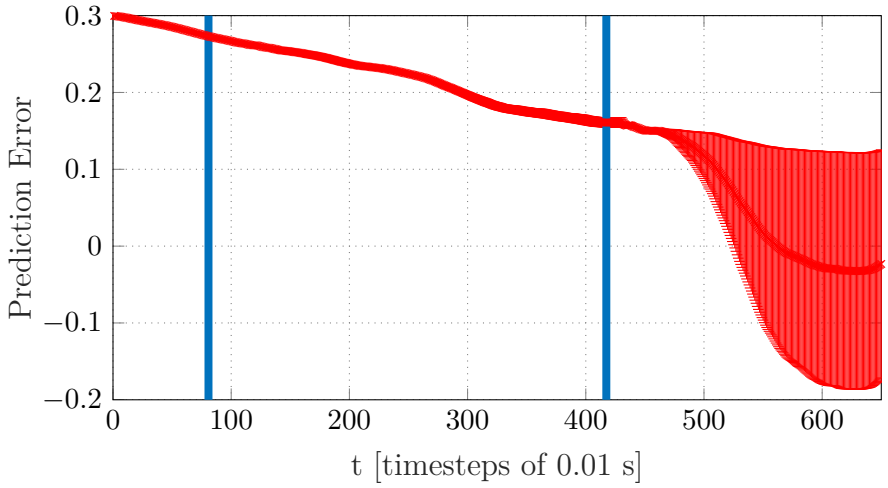


Figure 16: Prediction error bar for state prediction using phase 1 model parameters

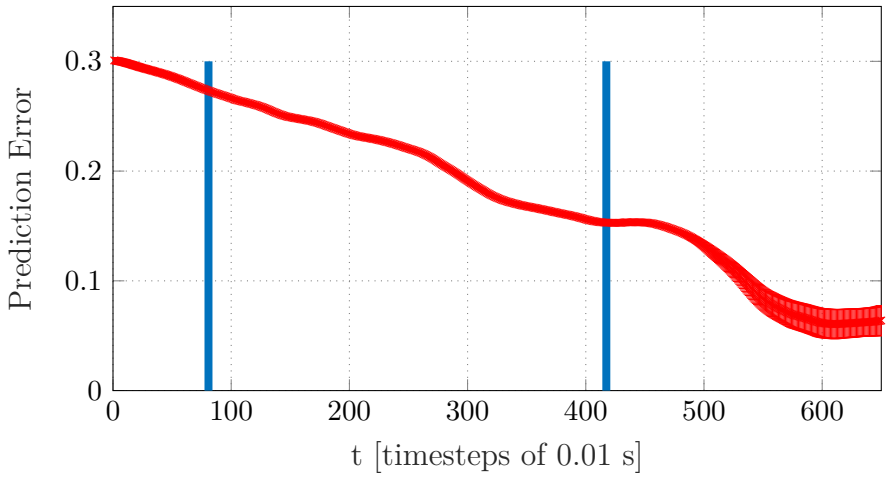


Figure 17: Prediction error bar for state prediction using phase 2 model parameters

one θ_1. The same conclusion can be made for state prediction using phase two and three.

6.2 Model parameter Estimation for two Demonstrations with Different Starting Points

To learn model parameters from multiple demonstrations, we used two demonstrations with different starting positions. The setup and the number of phases of the manipulation task were similar with the single demonstration setup as shown in Figure 12a and 12b. We used two demonstrations which were symmetric in z-axis in the path they follow. The x, y and z position of the tool for the two different starting positions are shown in Figure 21a and 21b and their corresponding Cartesian force is shown in Figure 24a and 24b. The symmetry of the demonstrated task can be seen Figure 21a and 21b in which the x and z components are similar, and the y

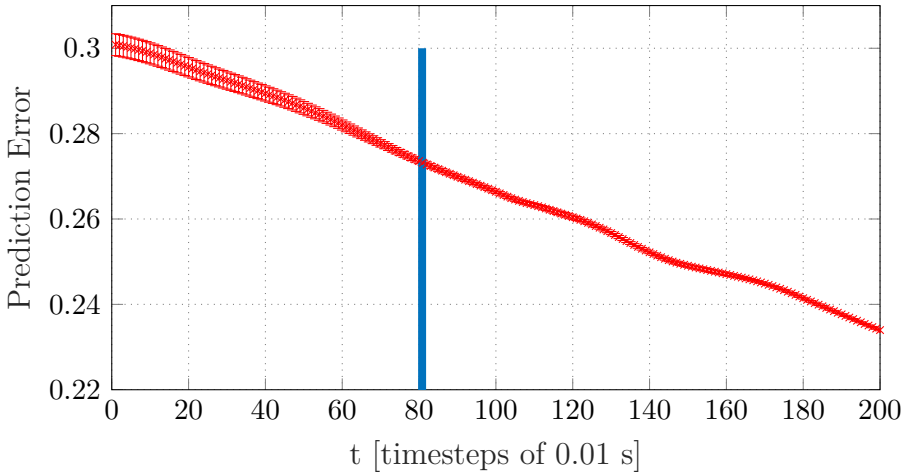


Figure 18: Prediction error bar for state prediction using phase 2 model parameters

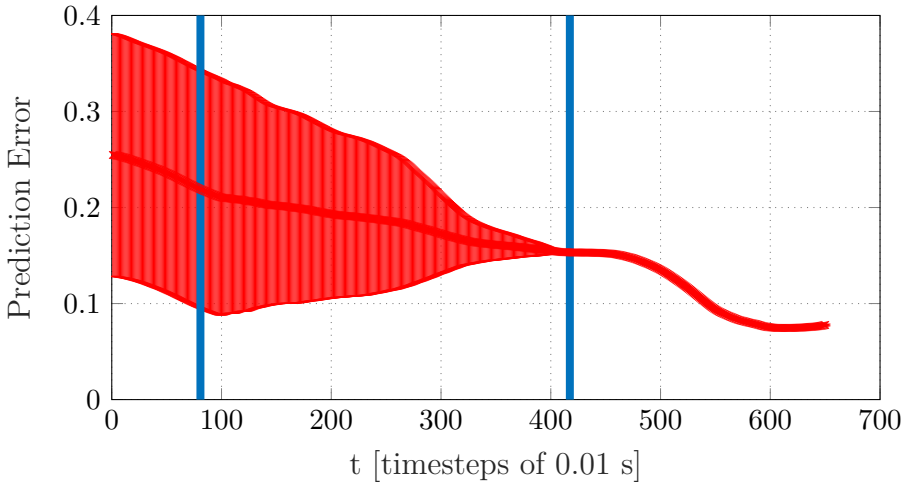


Figure 19: Prediction error bar for state prediction using phase 3 model parameters

component of the first task is negative of the second task. Similarly, the feature of the first demonstration is negative of the second demonstration.

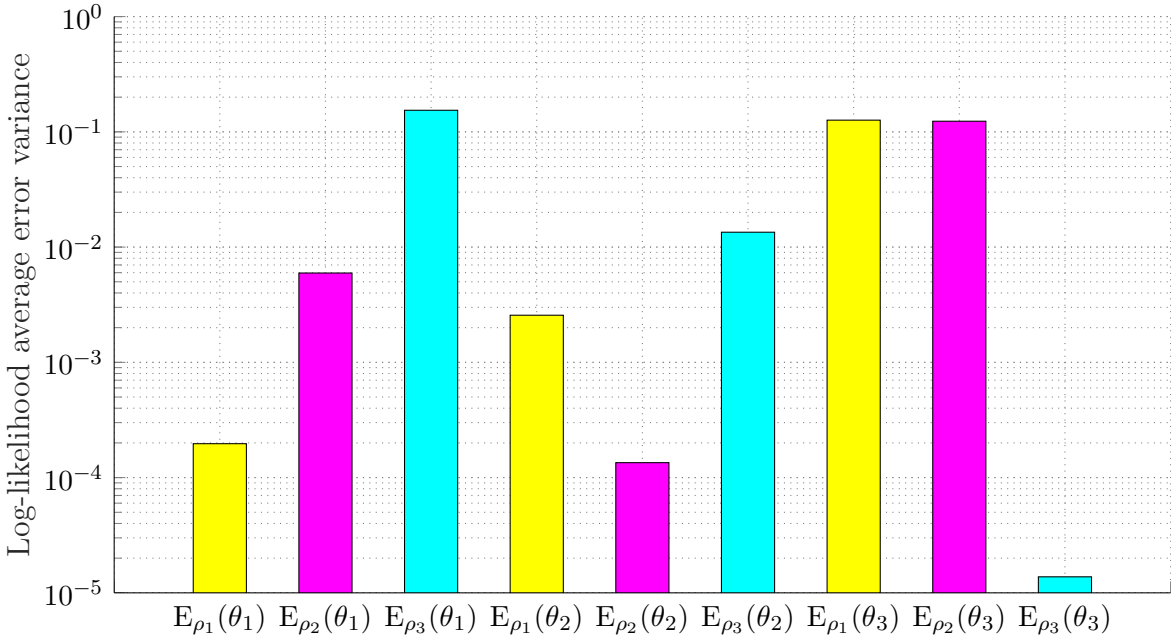


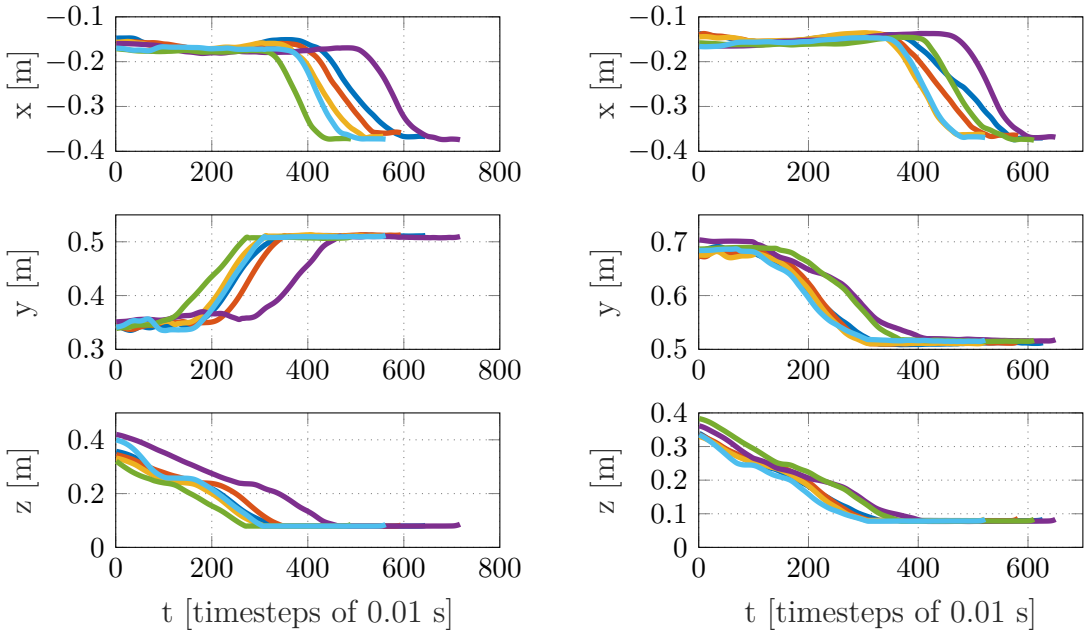
Figure 20: Prediction error bar for state prediction using phase 3 model parameters

After we recorded the required information for the two demonstrations, BIC was used to find the combined model number of the two demonstration pair with different starting positions. As it is pointed out in the single demonstration case, we found that the model number of the combined HMM model was the same as the model number estimated from single demonstration as shown in Figure 22

For estimating combined model parameters of the estimated model number, we ran the modified version of EM algorithm described in Section 4.5. To check if the implemented EM algorithm works we first used two similar demonstrations in order estimate the model parameters. The estimated phases sequence corresponding to the two concatenated demonstrations is shown in Figure 23. As indicated in the figure, the algorithm identifies the start of the next demonstration. Also the estimation works well as clearly shown by the approximately equal length of phases for both demonstrations and approximately equal model parameters for both cases. The comparison of model parameters estimated for the combined case with model parameters obtained from single demonstration was made as shown in Table C1. The model parameter obtained were approximately equal.

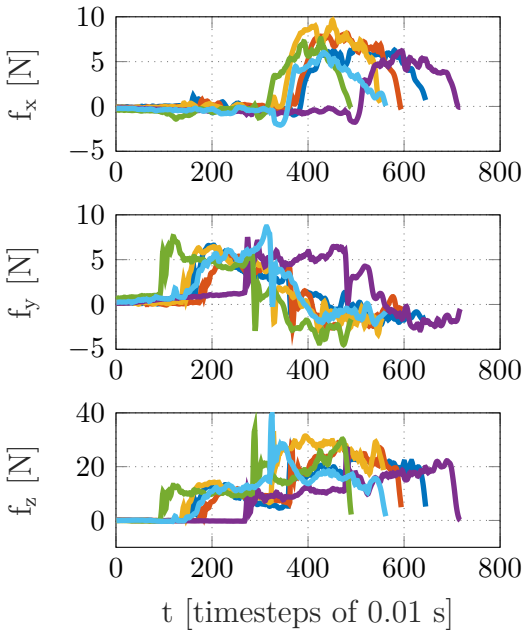
We tried two demonstrations with different starting position to estimate combined model parameters that generalize our model for both sides of our setup. As in the case of estimation from a single demonstration, the algorithm is able to estimate the phases of the assembly task as shown in Figure 15a which corresponds to the combined normalized forward variable of the HMM α. And the marked 3D plot of the position of the tool corresponding to the estimated phase sequence Figure 24a is shown in Figure 24b

To explain why the system behaves as it should, the time-varying transitions between the hidden phases were captured during reproduction. The plot with its

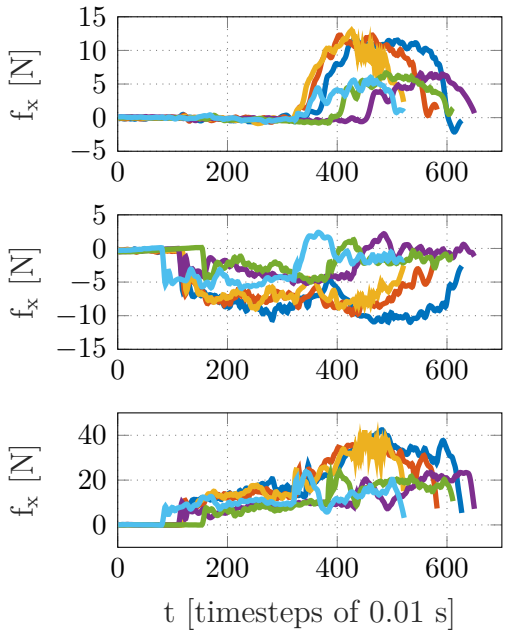


(a) x, y and z position of tool for SP 1

(b) x, y and z position of tool for SP 2



(c) Contact force for SP 1



(d) Contact force for SP 2

Figure 21: Position and contact force for two demonstration with different SP

corresponding feature vector and the estimated phase sequence is plotted side by side for reproduction phase as shown in Figure 25. The time varying transition from initial phase to the rest of the phases of the manipulation as clearly indicated in Figure 25a was executed smoothly which is shown by the only transition from the initial phase to subsequent phase. The phase transition change occurs when

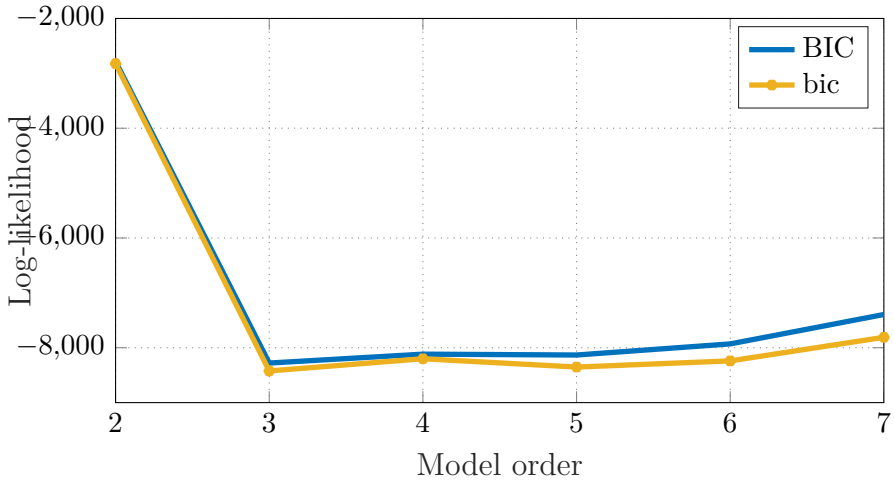


Figure 22: BIC based estimated model number for two demonstrations with different SPs

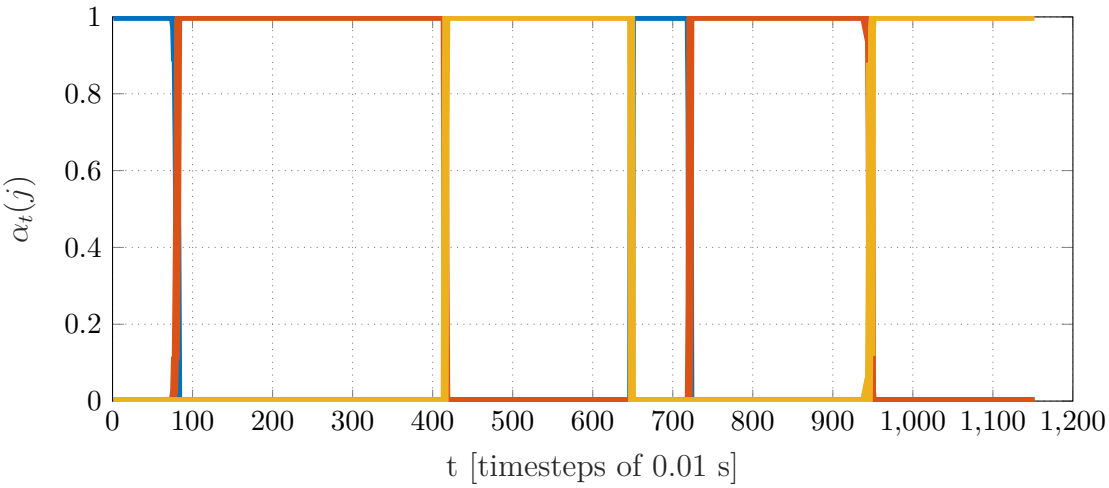
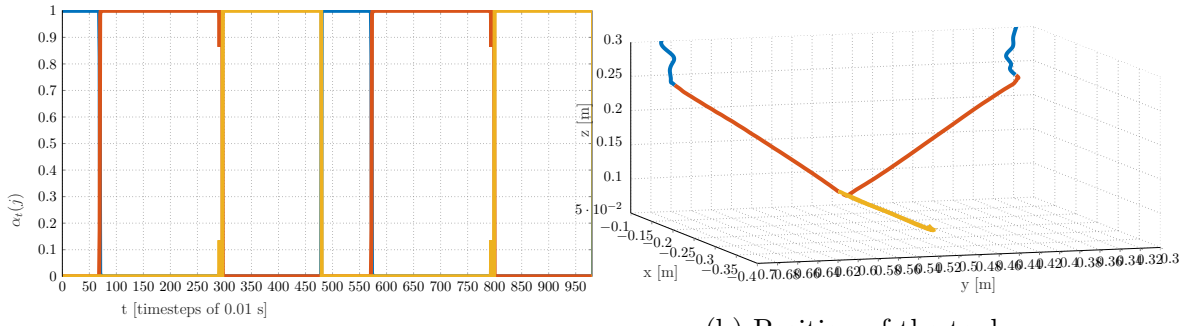


Figure 23: Estimated phases of the target manipulation task for two concatenated demonstrations (normalized α)

there is a change in the feature vector i.e. the contact force as indicated in Figure 25b. The time-varying phase transition nature with respect to the feature vector corresponds to the soft-max hidden phase transition definition of Section 4.1. The phase sequence which corresponds to the normalized forward variable is also plotted to show switching between the controller assigned for each phase as shown in Fig. 25c.

6.3 Comparison of STAR and ETAR

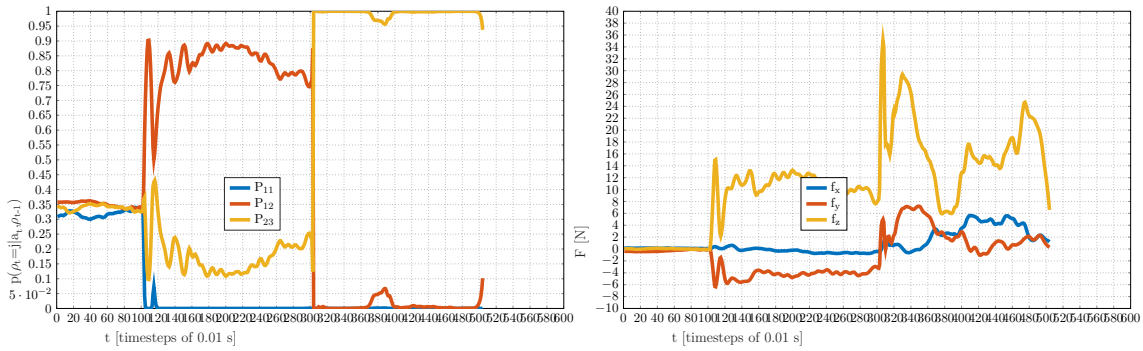
For showing the prediction accuracy, our approach was compared with the state based phase transition [7]. The comparison was made between effect-of-action based phase transition as shown in Figure 26a with the state based transition as shown in



(a) Estimated phases (normalized α)

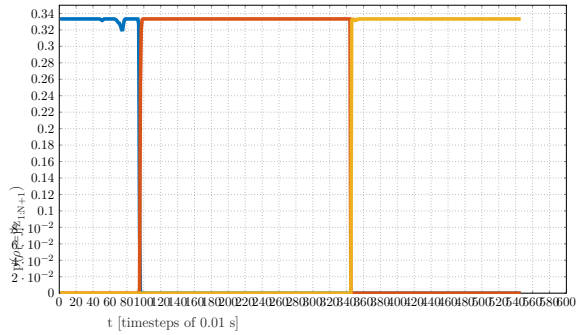
(b) Position of the tool

Figure 24: Estimated phases and corresponding position of tool for the manipulation task with different starting points



(a) Hidden phase transition probability

(b) Feature vector

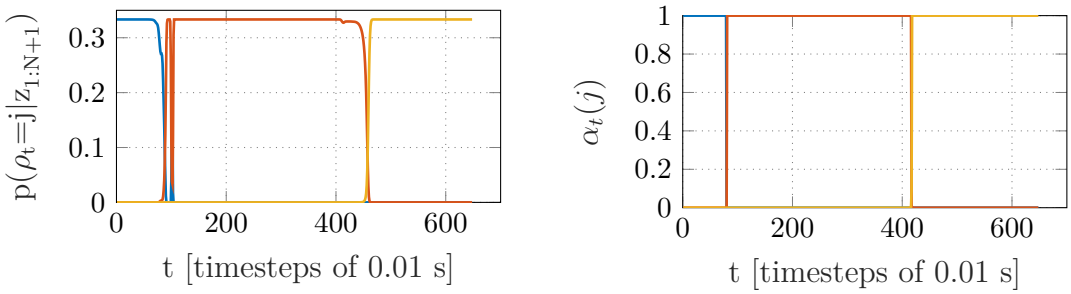


(c) Normalized forward variable

Figure 25: Hidden phase, feature vector and normalized forward variable

Figure 26b. For the state based transition, we took the relative position of the tool to a target (end) position as a feature vector. As shown in Figure 26a and Figure 26b a smooth phase transition is achieved when forces were considered as feature vector compared to the state based transition.

In addition, the state based estimation of model parameters results in inaccurate prediction of state compared to force based transitions especially during transition between phases.



(a) State based estimated phases

(b) force based estimated phases

Figure 26: Estimated phases sequence based on state based and effect-of-action (force) based phase transitions

6.4 Model parameter Estimation for Hose Coupler

As the final part of our experiment to make sure our proposed implementation can also work for setups that include rotation as a phase of the manipulation task, we use the hose coupler experimental setup as shown in Figure 27. The experimental setup includes two phases; the first phase is the unconstrained motion in which the hose tool moves toward the coupler. The second phase starts when the hose touches the coupler which then continues with rotational motion and ends when the hose-coupler is coupled. In this experiment since one of the two phases includes a rotational motion the state represents position and orientation instead of position only. Also, the feature vector is represented by concatenated force, torque and a scalar one.

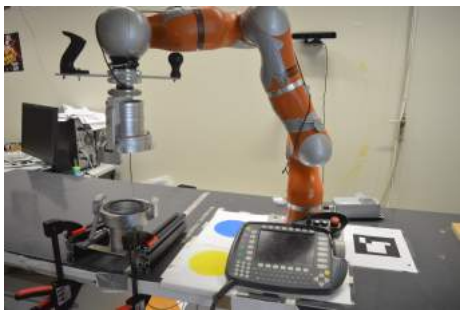
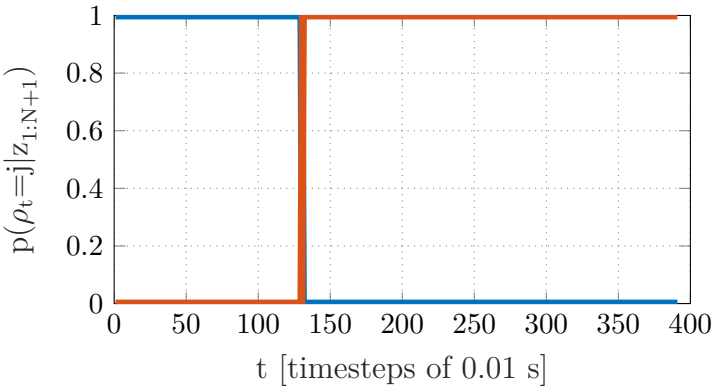


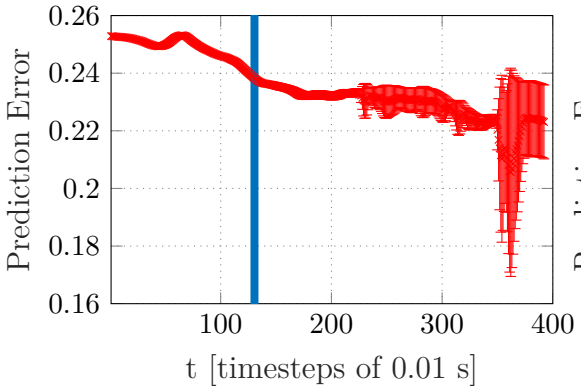
Figure 27: Hose coupler experimental setup

For the recorded data of the hose coupler setup of Fig. 27, BIC model selection criteria was run similarly as Valley setup case and we obtained a model order of $N = 2$. For the estimated model order EM algorithm for the modified state and feature definition was ran, as shown in Figure 28a, the algorithm was able to find actual phase sequence. To check if our model parameters estimated can predict we followed a similar procedure as the Valley case and average error bar corresponding to predictions from model parameters of phase one and two are shown in Figure 28b and 28c respectively. As shown in the Figure 28b the model parameters for phase one were able to predict next state of the first phase as lower error shows before the horizontal line. Similarly, the model parameters of phase two were able to predict

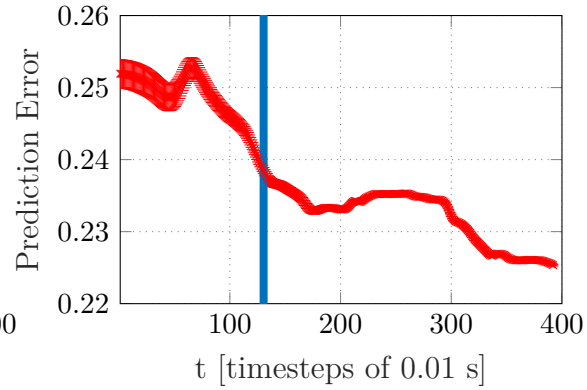
the second phase as shown with lower prediction error after the horizontal line of Figure 28c.



(a) Normalized forward variable



(b) Prediction based on phase one



(c) Prediction based on phase two

Figure 28: Estimated phase sequence and prediction using model parameters of phase one and two

The same procedure as Valley setup was followed for estimating the average error variance qualitatively. As shown in Figure 29, the prediction using the non-contact estimated model parameters predicted the next state of the initial phase with lower error labeled with $E_{\rho_1}(\theta_1)$ compared to the coupling phase labeled with $E_{\rho_2}(\theta_1)$. A similar conclusion can be made for the second phase prediction using the second phase model parameters.

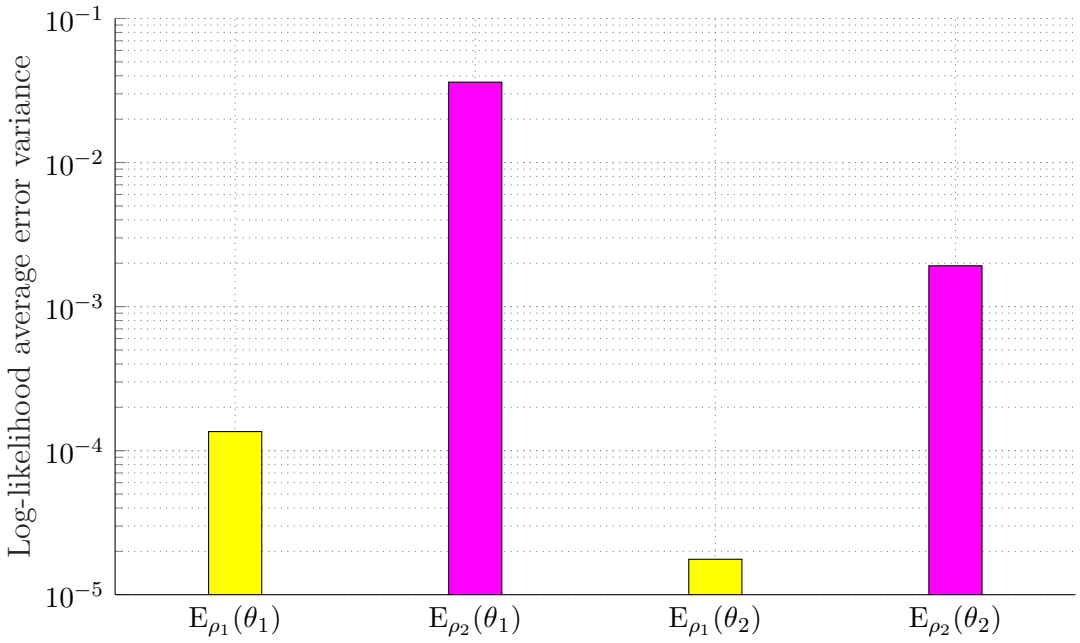


Figure 29: Prediction error bar for state prediction using phase 3 model parameters

7 Conclusion and Future Work

We presented an LfD approach based on a probabilistic modeling of a compliant motion based manipulation tasks with multiple phases. The phases were represented as hidden variables corresponding subgoals of the task. We focus on how the model parameters could be learned from a single and multiple demonstrations using the expectation-maximization algorithm for various definition of state and feature vectors. More specifically we propose a more generalized model for two different demonstrations. The proposed model incorporates the observed effect of performing an action when predicting the transitions between the hidden phases. Experiments evaluate the performance of the approach, showing that the off-line estimated model parameters are able to detect the current phase and transition between phases in which a defined controller assigned to each subtask is activated during the reproduction phase. The results showed that the effect of performing an action-based transitioning allows the robot to predict the phase changes more accurately than state based phase transition, resulting in better predictions overall.

The effect-of-action based transitions AR-HMM (ETAR) has feature vectors which are approximately constant for each phase of the task compared to STAR that makes it accurate phase transition prediction and robust against possible environment variations.

Our implemented algorithm can be used for any general manipulation task as it is proved for the hose coupler experimental setup with different effect-of-action and state definition that includes force, torque, position and orientation.

For future work a generalization, our work for practical application such as underwater assembly can be done by incorporating a visualization method to lead the tool to the target position and then our proposed method can be used for the final assembly task. In a case of better estimation of parameters requirement that can be used for general any complex assembly task modified model that can incorporate friction between the tool and the environment can be included for future work.

References

[1] De Schutter, Joris, and Hendrik Van Brussel. *Compliant robot motion I. A formalism for specifying compliant motion tasks.* The International Journal of Robotics Research 7.4 (1988): 3-17.

[2] Kormushev, Petar, Sylvain Calinon, and Darwin G. Caldwell. *Reinforcement learning in robotics: Applications and real-world challenges.* Robotics 2.3 (2013): 122-148.

[3] Rozo, Leonel, Pablo Jiménez, and Carme Torras. *Robot learning from demonstration of force-based tasks with multiple solution trajectories.* Advanced Robotics (ICAR), 2011 15th International Conference on. IEEE, 2011.

[4] Lee, Dongheui, and Christian Ott. *Incremental kinesthetic teaching of motion primitives using the motion refinement tube.* Autonomous Robots 31.2-3 (2011): 115-131.

[5] Marcia Riley, Ales Ude, Christopher Atkeson Gordon Cheng. *Coaching: An Approach to Efficiently and Intuitively Create Humanoid Robot Behaviors.* Humanoid Robots, 2006 6th IEEE-RAS International Conference, 2006.

[6] Suomalainen, Markku, Kyrki, Ville. *Learning compliant assembly motions from demonstration.* Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems Pages 4326-4332.

[7] Kroemer, Oliver, Van Hoof, Herke Neumann, Gerhard Peters, Jan *Learning to predict phases of manipulation tasks as hidden states* Proceedings - IEEE International Conference on Robotics and Automation Pages 4009-4014.

[8] Billard, Aude, Sylvain Calinon, Ruediger Dillmann, and Stefan Schaal. *Robot programming by demonstration.* In Springer handbook of robotics, pp. 1371-1394. Springer Berlin Heidelberg, 2008.

[9] Kemp, Charles C., Aaron Edsinger, and Eduardo Torres-Jara. *Challenges for robot manipulation in human environments.* IEEE Robotics and Automation Magazine, 14.1:20-29, 2007.

[10] Flanagan, J. Randall, Miles C. Bowman, and Roland S. Johansson. *Control strategies in object manipulation tasks.* Current opinion in neurobiology 16.6 (2006): 650-659.

[11] Argall, Brenna D., Sonia Chernova, Manuela Veloso, and Brett Browning. *A survey of robot learning from demonstration.* Robotics and autonomous systems 57.5 (2009): 469-483.

[12] Montebelli, Alberto, Franz Steinmetz, and Ville Kyrki. *On handing down our tools to robots: Single-phase kinesthetic teaching for dynamic in-contact tasks.* Robotics and Automation (ICRA), 2015 IEEE International Conference on. IEEE, 2015.

[13] Akgun, Baris, and Kaushik Subramanian. *Robot learning from demonstration: kinesthetic teaching vs. teleoperation.* Unpublished manuscript (2011).

[14] Calinon, Sylvain, and Aude Billard. *Active teaching in robot programming by demonstration.* Robot and Human interactive Communication, 2007. RO-MAN 2007. The 16th IEEE International Symposium on. IEEE, 2007.

[15] Wolpert, Daniel M., Zoubin Ghahramani, and Michael I. Jordan. *An internal model for Sensorimotor integration* Science; Washington 269.5232 Sep. 29, 1995; page 1880.

[16] C. Sutton, A. McCallum *An Introduction to Conditional Random Fields for Relational Learning, in Introduction to Statistical Relational Learning.* L. Getoor and B. Taskar, eds., the MIT Press, 2007.

[17] Tang Xuan. *Autoregressive Hidden Markov Model with Application in an El Ñino Study.* Diss. Master Thesis, University of Saskatchewan, 2004.

[18] Kevin, P. Murphy. *Switching Kalman Filters* Dept. of Computer Science, University of California, August 1998.

[19] Bicchi, Antonio, J. Kenneth Salisbury, and David L. Brock. *Contact Sensing from Force Measurements.* The International Journal of Robotics Research 12.3 (1993): 249-262.

[20] A.O. Farahat, B.S. Graves, J.C. Trinkle. *Identifying Contact Formations in the Presence of Uncertainty.* Intelligent Robots and Systems 95. 'Human Robot Interaction and Cooperative Robots', Proceedings. 1995 IEEE/RSJ International Conference.

[21] Eberman, Brian, and J. Kenneth Salisbury. *Application of Change Detection to Dynamic Contact Sensing.* The International Journal of Robotics Research 13.5 (1994): 369-394.

[22] McCarragher, Brenan J., and Haruhiko Asada. *Qualitative Template Matching Using Dynamic Process Models for State Transition Recognition of Robotic Assembly.* Journal of Dynamic Systems, Measurement, and Control 115.2A (1993): 261-269.

[23] Wim Meeussen, Johan Rutgeerts, Klaas Gadeyne, Herman Bruyninckx, Member, IEEE, and Joris De Schutter, Member, IEEE. *Contact-State Segmentation Using Particle Filters for Programming by Human Demonstration in Compliant-Motion Tasks.* IEEE TRANSACTIONS ON ROBOTICS, VOL. 23, NO. 2, APRIL 2007

[24] Wim Meeussen. *Compliant Robot Motion: from Path Planning or Human Demonstration to Force Controlled Task Execution.* Katholieke Universiteit Leuven, Faculteit Toegepaste Wetenschappen, Arenbergkasteel, B-3001 Heverlee (Leuven), Belgium. ISBN 978-90-5682-753-3, 21 December 2006

[25] K. Gadeyne T. Lefebvre H. Bruyninckx.*Bayesian Hybrid Model-State Estimation Applied to Simultaneous Contact Formation Recognition and Geometrical Parameter Estimation* The International Journal of Robotics Research August 1, 2005

[26] A. Denasi, B. T. Verhaar, D. Kosti, D.J.H. Bruijnen, H. Nijmeijer. *Robot Programming by Demonstration.* In Proceedings of the 10th Philips Conference on Applications of Control Technology, 3-4 February 2009, Hilvarenbeek, The Netherlands (pp. 149-153). Netherlands, Hilvarenbeek: Technische Universiteit Eindhoven

[27] Petar Kormushev, Sylvain Calinon & Darwin G. Caldwel. *Imitation Learning of Positional and Force Skills Demonstrated via Kinesthetic Teaching and Haptic Input.* Advanced Robotics, 25:5, 581-603.

[28] Stefano Cabras, María Eugenia Castellanos, Ernesto Staffett. *Contact-State Classification in Human-Demonstrated Robot Compliant Motion Tasks Using the Boosting Algorithm* IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBER NETICS-PART B: CYBERNETICS, VOL. 40, NO. 5, October 2010

[29] Thomas J. Debus, Pierre E. Dupont, Robert D. Howe. *Contact State Estimation using Multiple Model Estimation and Hidden Markov Models* The International Journal of Robotics Research April 01, 2004.

[30] Geir E. Hovland, Brenan J. McCarragher. *Hidden Markov Models as a Process Monitor in Robotic Assembly* The International Journal of Robotics Research February 01, 1998.

[31] G.E. Hovland, P. Sikka B. J. McCarragher. *Skill acquisition from human demonstration using a hidden Markov model* Proceedings of the 1996 IEEE International Conference on Robotics and Automation Minneapolis, Minnesota - April 1996.

[32] H.Y.K. Lau. *A hidden Markov model-based assembly contact recognition system* Mechatronics 13 (2003) 1001-1023.

[33] S. Andrews, P.G. Kry. *Goal directed multi-finger manipulation: Control policies and analysis.* Computers and Graphics, 37(7):830 – 839, 2013.

[34] Joseph Romano, Kaijen Hsiao, Günter Niemeyer, Sachin Chitta, and Katharine J. Kuchenbecker. *Human-inspired robotic grasp control with tactile sensing.* IEEE Transactions on Robotics, 27:1067-1079, 2011.

[35] Roland S. Johansson and Randall J. Flanagan. *Coding and use of tactile signals from the fingertips in object manipulation tasks.* Nature reviews. Neuroscience, 10(5):345-359, April 2009.

[36] Rabiner, Lawrence R. *A tutorial on hidden Markov models and selected applications in speech recognition*. Proceedings of the IEEE 77.2 (1989): 257-286.

[37] Servitja Robert, Maria. *A first study on Hidden Markov Models and one application in speech recognition*. (2016). http://www.diva-portal.org/smash/record.jsf?pid=diva2%3A893683&dswid=169#sthash.i25fHwUs.dpbs

[38] Read, Jonathon. *Hidden Markov Models and Dynamic Programming*. University of Oslo (2011).

[39] Mitchell, Tom M. *Machine Learning*. McGraw-Hill International Editions Computer Science Series. (1997).

[40] Anzai, Yuichiro. *Pattern recognition and machine learning*. Elsevier, 2012.

[41] Jebara, Tony. *Machine learning: discriminative and generative*. Vol. 755. Springer Science and Business Media, 2012.

[42] Martin Tykal. *Optimizing Programming by Demonstration for In-contact Task Models by Incremental Learning*. Master's thesis. Aalto University, School of Electrical Engineering, 2015.

[43] Rainer Bischoff et al. *The KUKA-DLR Lightweight Robot arm-a new reference platform for robotics research and manufacturing*. In: Robotics (ISR), 2010 41st international symposium on and 2010 6th German conference on robotics (ROBOTIK).VDE. 2010, pp. 1-8

[44] ATI Industrial Automation. *Six-Axis Force/Torque Transducer*. 9620-05-transducer section - 17 edition. Apex, NC, USA, Apr. 2013.

[45] Mattia Racca. *Programming by Demonstration using Hidden Semi-Markov Models*. Master's thesis. Aalto University, School of Electrical Engineering, 2015.

[46] Gunter Schreiber, Andreas Stemmer, and Rainer Bischoff. *The fast research interface for the kuka lightweight robot*. In: IEEE Workshop on Innovative Robot Control Architectures for Demanding (Research) Applications How to Modify and Enhance Commercial Controllers (ICRA 2010). 2010

[47] http://cs.stanford.edu/people/tkr/fri/html/index.html

[48] KUKA Laboratories GmbH. *KUKA System Software 5.6 lr: Operating and Programming Instructions for System Integrators*. KSS 5.6 lr SI V5 en. Jan. 2012

[49] Bruno Siciliano and Oussama Khatib. *Springer handbook of robotics*. Springer Science & Business Media, 2008

[50] Chien, Jen-Tzung, and Sadaoki Furui. *Predictive hidden Markov model selection for speech recognition.* IEEE Transactions on Speech and Audio Processing 13.3 (2005): 377-387.

[51] H. Linhart, W. Zucchini. *Model Selection.* Edition, illustrated. Publisher, Wiley, 1986. Original from, University of Minnesota. Digitized, Jan 22, 2010. ISBN, 0471837229, 9780471837220.

[52] Walter Zucchini, Iain L. MacDonald, Roland Langrock. *Hidden Markov Models for Time Series: An Introduction Using R.* Chapman & Hall/CRC Monographs on Statistics & Applied Probability, April 28, 2009, ISBN 9781420010893 - CAT# CE5734

[53] L Wasserman. *Bayesian model selection and model averaging.* Journal of Mathematical Psychology, 44 (2000), pp. 92-107

[54] Ruppert, David, and Matthew P. Wand. *Multivariate locally weighted least squares regression.* The annals of statistics (1994): 1346-1370.

[55] http://eli.thegreenplace.net/2016/the-softmax-function-and-its-derivative/

[56] Kelley, Carl T. *Iterative methods for optimization.* Society for Industrial and Applied Mathematics, 1999.

[57] (KUKA) https://www.kuka.com/

[58] (DLR) www.dlr.de/rm/en

A Forward and Backward Variable Derivation

Using the forward variable definition, conditional independence and Bayes rule, the forward variable can be derived as follows

$$\begin{aligned}
\alpha_j(t) &= p(\rho_t = j) = p(z_{1:t+1}, \rho_t = j) \\
&= p(z_{1:t+1}|\rho_t = j)p(\rho_t = j) \\
&= p(z_{t+1}|\rho_t = j, z_t)p(z_{1:t}|\rho_t = j)p(\rho_t = i) \\
&= p(z_{t+1}|\rho_t = j, z_t)p(z_{1:t}, \rho_t = j) \\
&= p(z_{t+1}|\rho_t = j, z_t)\sum_i p(z_{1:t}, \rho_{t-1} = i, \rho_t = j) \\
&= p(z_{t+1}|\rho_t = j, z_t)\sum_i p(z_{1:t}, \rho_t = j|\rho_{t-1} = i)p(\rho_{t-1} = i) \\
&= p(z_{t+1}|\rho_t = j, z_t)\sum_i p(z_{1:t}|\rho_{t-1} = i)p(\rho_t = j|\rho_{t-1} = i)p(\rho_{t-1} = i) \\
&= p(z_{t+1}|\rho_t = j, z_t)\sum_i p(z_{1:t}, \rho_{t-1} = i, \rho_t = j)p(\rho_t = j|\rho_{t-1} = i) \\
&= p(z_{t+1}|\rho_t = j, z_t)\sum_i \alpha_j(t-1)p(\rho_t = j|\rho_{t-1} = i)
\end{aligned}$$

Similarly using the backward variable definition, conditional independence and Bayes rule, the backward variable can be derived as follows

$$\begin{aligned}
\beta_j(t) &= p(\rho_t = j) = p(z_{t+1:N}|\rho_t = j, z_{t+1}) \\
&= \sum_i p(z_{t+1:N}, \rho_{t+1}|\rho_t = i, z_{t+1}) \\
&= \sum_i p(z_{t+1:N}|z_{t+1}, \rho_{t+1} = i|\rho_t = j)p(\rho_{t+1} = i|\rho_t = j) \\
&= \sum_i p(z_{t+1:N}|z_{t+1}, \rho_{t+1} = i)p(\rho_{t+1} = i|\rho_t = j) \\
&= \sum_i p(z_{t+2:N}|\rho_{t+1} = i)p(z_{t+1}|\rho_t = i, z_t)p(\rho_{t+1} = i|\rho_t = j) \\
&= \sum_i \beta(\rho_{t+1} = i)p(z_{t+1}|\rho_t = i, z_t)p(\rho_{t+1} = i|\rho_t = j)
\end{aligned}$$

B Expectation-Maximization Algorithm (Derivation)

The target of EM algorithm is maximizing log-likelihood of the observation data $\log(p(\mathbf{z}))$. But maximizing log-likelihood of the observation data is not tractable, which means there is closed form solution.

Instead we introduce hidden variables to make it tractable. The probability of the observed data in terms of the hidden variable is given by using Baye's rule

$$p(\mathbf{z}) = \frac{p(\mathbf{z}, \rho)}{p(\rho|\mathbf{z})}$$

its corresponding log-likelihood is given by

$$\log(p(\mathbf{z})) = \log(p(\mathbf{z}, \rho)) - \log(p(\rho|\mathbf{z}))$$

adding and subtracting the same variable will result

$$\log(p(\mathbf{z})) = \log(p(\mathbf{z}, \rho)) - \log(q(\rho)) - \log(p(\rho|\mathbf{z})) + \log(q(\rho)) = \log \frac{p(\mathbf{z}, \rho)}{q(\rho)} + \left(- \log \frac{p(\rho|\mathbf{z})}{q(\rho)} \right)$$

multiplying both sides by $q(\rho)$ results in

$$q(\rho) \log(p(\mathbf{z})) = q(\mathbf{z}) \log \frac{p(\mathbf{z}, \rho)}{q(z)} + q(z) \left(- \log \frac{p(z/x)}{q(z)} \right)$$

Integrating both sides with respect to $d\rho$

$$\int q(\rho) \log(p(\mathbf{z})) d\rho = \int q(\rho) \log \frac{p(\mathbf{z}, \rho)}{q(\rho)} d\rho - \int q(\rho) \log \frac{p(\rho|\mathbf{z})}{q(\rho)} d\rho = F(q, \theta) + KL(p|q)$$

The second term in the right hand side of the equation is the expectation of $\log \frac{p(\rho|\mathbf{z})}{p(\mathbf{z})}$ under the probability distribution $q(\rho)$ and is known as Kullback-Leibler (KL) divergence. The term $KL(p|q)$ is always greater than 0. Since $KL(p|q) \geq 0$, $F(q, \theta)$ becomes strictly lower bounded on log-likelihood of the observed data.

Generally the purpose of the EM algorithm is make $q(\rho)$ as close as possible to $p(\rho|\mathbf{z})$ and maximize the lower bound $F(q, \theta)$ with respect to the model parameter. This done in two steps the expectation step (E-step) and the maximization step (M-step).

1. E-step:

   make $q(\rho)$ as close as possible to $p(\rho|\mathbf{z})$

   $$q(\rho) \approx p(\rho|\mathbf{z}) \tag{B1}$$

2. M-step:

   Maximizing lower bound $F(q, \theta)$ with respect to model parameters θ

   $$max_\theta F(q, \theta) \tag{B2}$$

when maximizing $F(q, \theta)$, q becomes different with p and we need to go back to E-step to make close $q(\rho) \approx p(\rho|\mathbf{z})$

$$F(q, \theta) = \int q(\rho) \log \frac{p(\mathbf{z}, \rho)}{q(\rho)} d\rho = \int q(\rho) \log p(\mathbf{z}, \rho) d\rho - \int q(\rho) \log q(\rho) d\rho$$

$q(\rho)$ is independent of the model parameter θ, therefore maximizing $F(q, \theta)$ is equivalent to

$$max_\theta F(q, \theta) = max_\theta \int q(\rho) \log p(\mathbf{z}, \rho) dz \tag{B3}$$

C Parameter Estimated Comparison for Two Similar Demonstrations

	Demo. 1	**Demo. 2**
w	$\begin{pmatrix} -0.0718 & 0.2224 & -0.7329 & -0.2044 \\ -0.1365 & 1.2335 & -3.5037 & -0.5463 \\ -3.4286 & 4.3417 & -25.5330 & -2.1910 \end{pmatrix}$ $\begin{pmatrix} -0.1017 & -0.1490 & 0.1187 & -0.1101 \\ -0.1721 & 0.7842 & -2.4817 & -0.4314 \\ -3.4891 & 3.5624 & -23.7947 & -1.9888 \end{pmatrix}$ $\begin{pmatrix} 0.2391 & 0.2146 & 0.5619 & -0.1566 \\ 0.1663 & 1.1163 & -1.9635 & -0.4747 \\ -2.7362 & 4.4064 & -22.7675 & -2.0791 \end{pmatrix}$	$\begin{pmatrix} -0.1713 & 0.2406 & -0.9238 & -0.6807 \\ -8.6553 & 6.6203 & -100.4144 & -8.0383 \\ -0.0844 & 0.0330 & -0.7543 & -0.1498 \end{pmatrix}$ $\begin{pmatrix} -0.0449 & 0.0312 & -0.3236 & -0.1209 \\ -0.1777 & 0.2804 & -0.8905 & -0.7561 \\ -9.2343 & 2.2641 & -91.5347 & -7.2839 \end{pmatrix}$ $\begin{pmatrix} -5.8055 & 9.5197 & -83.4395 & -7.3135 \\ -0.0197 & 0.0980 & -0.2985 & -0.1554 \\ -0.1981 & 0.3496 & -1.3217 & -0.8315 \end{pmatrix}$
A	$\begin{pmatrix} 0.9807 & 0.0082 & -0.0011 \\ 0.0086 & 0.9574 & -0.0053 \\ 1.4011 & 0.0997 & 1.0066 \end{pmatrix}$ $\begin{pmatrix} 0.9776 & -0.0001 & -0.0019 \\ 0.1193 & 0.9547 & 0.0585 \\ 0.0796 & 0.0149 & 0.9944 \end{pmatrix}$ $\begin{pmatrix} 1.0133 & -1.3959 & 0.1248 \\ 0.0002 & 0.9941 & 0.0988 \\ -0.0001 & 0.0433 & 1.1840 \end{pmatrix}$	$\begin{pmatrix} 0.9853 & -0.0016 & -0.0006 \\ -0.3154 & 1.0290 & 0.0063 \\ 0.2408 & 0.0045 & 0.9983 \end{pmatrix}$ $\begin{pmatrix} 0.9720 & -0.0665 & 0.0602 \\ -0.0462 & 1.1060 & -0.1106 \\ -0.0702 & 0.2020 & 0.7935 \end{pmatrix}$ $\begin{pmatrix} 0.9871 & 0.1002 & 0.2423 \\ -0.0004 & 1.0167 & 0.0453 \\ 0.0004 & 0.0681 & 1.2284 \end{pmatrix}$
B	$\begin{pmatrix} -0.0000 & -0.0001 & 0.0002 & -0.0086 \\ 0.0008 & -0.0012 & -0.0006 & 0.0325 \\ 0.0009 & -0.0013 & -0.0009 & 0.1577 \end{pmatrix}$ $\begin{pmatrix} -0.0001 & -0.0001 & -0.0000 & -0.0030 \\ -0.0003 & 0.0003 & 0.0001 & 0.0351 \\ -0.0003 & 0.0002 & 0.0001 & 0.0032 \end{pmatrix}$ $\begin{pmatrix} -0.0005 & -0.0002 & 0.0000 & 0.7163 \\ -0.0000 & 0.0000 & -0.0000 & -0.0045 \\ -0.0000 & 0.0000 & -0.0000 & -0.0367 \end{pmatrix}$	$\begin{pmatrix} 0.0002 & -0.0000 & -0.0000 & -0.0012 \\ 0.0000 & 0.0002 & 0.0000 & -0.0745 \\ -0.0001 & -0.0000 & 0.0000 & 0.0367 \end{pmatrix}$ $\begin{pmatrix} 0.0000 & -0.0000 & 0.0000 & 0.0256 \\ 0.0003 & 0.0002 & -0.0001 & -0.0535 \\ 0.0003 & 0.0002 & -0.0001 & -0.0998 \end{pmatrix}$ $\begin{pmatrix} -0.0001 & 0.0003 & 0.0001 & -0.0773 \\ -0.0000 & 0.0000 & -0.0000 & -0.0121 \\ -0.0000 & 0.0000 & 0.0000 & -0.0528 \end{pmatrix}$
Σ	$10^{-9} \times \begin{pmatrix} 0.0119 & -0.0319 & -0.0499 \\ -0.0319 & 0.5800 & 0.4603 \\ -0.0499 & 0.4603 & 0.8842 \end{pmatrix}$ $10^{-7} \times \begin{pmatrix} 0.0409 & -0.0845 & -0.0757 \\ -0.0845 & 0.4565 & 0.4003 \\ -0.0757 & 0.4003 & 0.4545 \end{pmatrix}$ $10^{-6} \times \begin{pmatrix} 0.2777 & 0.0044 & -0.0088 \\ 0.0044 & 0.0008 & -0.0003 \\ -0.0088 & -0.0003 & 0.0006 \end{pmatrix}$	$10^{-8} \times \begin{pmatrix} 0.0195 & -0.0066 & -0.0325 \\ -0.0066 & 0.0328 & -0.0354 \\ -0.0325 & -0.0354 & 0.6798 \end{pmatrix}$ $10^{-7} \times \begin{pmatrix} 0.0604 & -0.0390 & -0.0580 \\ -0.0390 & 0.8151 & 0.3356 \\ -0.0580 & 0.3356 & 0.8566 \end{pmatrix}$ $10^{-6} \times \begin{pmatrix} 0.1381 & -0.0016 & -0.0002 \\ -0.0016 & 0.0020 & -0.0005 \\ -0.0002 & -0.0005 & 0.0003 \end{pmatrix}$

Table C1: Model parameter comparison for two similar demonstrations